APPLICATIONS OF FLOATING-GATE BASED PROGRAMMABLE MIXED-SIGNAL RECONFIGURABLE SYSTEMS

A Dissertation
Presented to
The Academic Faculty

by

Farhan Adil

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2014

APPLICATIONS OF FLOATING-GATE BASED PROGRAMMABLE MIXED-SIGNAL RECONFIGURABLE SYSTEMS

Approved by:

Professor Jennifer Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Hua Wang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Eugenio Culurciello
School of Biomedical Engineering
Purdue University

Professor Omer Inan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 19 August 2014

ACKNOWLEDGEMENTS

First and foremost, I would like thank my adviser Dr. Jennifer Hasler for her wisdom and support throughout my graduate school career. This would not have been possible without her guidance over the years. I would also like to thank my committee members: Dr. Hua Wang, Dr. Omer Inan, Dr. Saibal Mukhopadhyay, and Dr. Eugenio Culurciello, for the time they have taken to serve on my committee, their insight on my research, and their feedback as well. I would, especially, like to thank all the wonderful and brilliant members of ICELAB both past and present. They created a very friendly and warm atmosphere throughout the years. I would like to thank my Mom, Dad, and my Sister, Ayesha, for their love and support. Last and most importantly, I would like to thank my wife, Lisa. Her patience, love, and encouragement meant everything to me. This journey would not have been possible without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A mixed-signal reconfigurable platform gives the designer the choice of implementing systems using the benefits of both analog and digital circuits. The subject of this research is the implementation and application of mixed-signal reconfigurable systems utilizing floating-gate transistors and field programmable analog/digital arrays.

Basic analog circuits using floating-gate CMOS devices have been developed for this research. Floating-gate based analog circuits reduce the effects of inherent property mismatch present in analog circuits. Various circuit blocks including current mirrors, gilbert multipliers, and $G_m - C$ filters were designed and experimentally demonstrated to show reduced mismatch effects. Such floating-gate transistors and circuits are the basis for the reconfigurable systems developed in this research.

To enable high-performance reconfigurable systems, sub-micron and sub-$100nm$ CMOS process nodes were used in this research. At such small process nodes, the scaling of Floating-gate devices is a key issue. Test structures were created to verify the programming capability for floating-gate devices at various process nodes. Experimental results show scalability of floating-gate devices along with effective charge programming ability.

A floating-gate based reconfigurable mixed-signal platform using Field-Programmable Array of Analog-Digital Devices (FPAADD) has been created and experimentally verified. Further FPAADD systems augmented with a CPU based digital back-end were developed to enable greater applications for such reconfigurable systems. Experimental functionality and circuits/systems created using FPAADD based systems were demonstrated for this research work.

CHAPTER I

RECONFIGURABLE SYSTEMS

Reconfigurable systems exist as an attractive alternative to custom ASIC design when the monetary cost of fabrication, or the manufacturing time is too high. Digital systems cater particularly well to reconfigurability in that any digitally solvable problem can be implemented with a very small number of building blocks that are functionally insensitive to fan-out and fan-in. FPGAs use look-up tables (LUTs) and flip flops to implement arbitrary and small number-of-input Boolean equations and state machines. High level functions are built up from large numbers of these blocks being connected together by a programmable interconnection network (the interconnect). With digital design's ability to be abstracted to very high level programming languages, systems can be rapidly prototyped and implemented on FPGAs. Of course this flexibility does not come without an associated cost increase to area, power, and degradation of system speed.

While all solvable problems can be solved in the digital domain, some problems map more efficiently to other domains. Problems like integer factorization, searching unsorted lists, and simulating quantum many-body systems, for instance, have solutions implementable on quantum computers that are algorithmically more efficient than the best known solutions on probabilistic Turing machines (classical digital computers). The filtering, smoothing, or modulation of sensor signals are efficiently solved in the domain of analog signal processing or analog computation. For a digital computer to even begin to work on real world data, some sort of analog processing must take place to convert it into a compatible format.

Analog solutions, when implemented in silicon, incur the same costs of fabrication

and design time iteration that makes reconfigurable solutions attractive for many applications. The FPAA is the analog equivalent of the FPGA. In essence it is a set of low level analog computational elements in a reconfigurable interconnect. Unlike FPGAs, however, the choice of computational elements tends to vary quite a bit, and thus FPAAs come in many different flavors: some use discrete-time, switched-capacitors, some are based on operational amplifiers and Gm-C circuits, some use translinear elements as the building blocks, and some everything in between [2, 3, 4, 5, 6, 7].

CHAPTER II

FLOATING-GATE TRANSISTOR DESIGN IN CMOS

FPAADD systems may be further enhanced by usage of floating-gate transistors as switch elements and in sub-circuits. A floating-gate device is a CMOS transistor for which the gate terminal is completely isolated to any DC signal path. Signals couple into the gate through an input capacitor, C_g, which can be either a poly-poly or MOS capacitor. The input capacitor provides the necessary electrical isolation to the gate terminal. Electrical isolation allows non-volatile charge storage at that same node [8]. With the ability to change the stored charge, the floating-gate transistor adds fine-grain reconfigurability to FPAADD systems. Such a reconfigurable system allows the reduction of analog sources of error, i.e. device mismatch. Reducing mismatch errors are key in obtaining high performance data converters. Analog solutions for reducing such errors are generally lower power compared to digital solutions, providing another benefit. We will show device mismatch is reduced using floating-gate devices. For example, the input offset voltage of amplifiers can be reduced greatly without the need for any external extra noisy elements, such as a chopper circuit. Additionally, floating-gate devices enable fine tuning and calibration of mixed signal systems, as well. We will present a floating-gate based filter with the ability to change the corner frequency and/or Q of a filter. Each floating-gate device may control one aspect of the filter, replacing calibration DACs in traditional systems.

2.1 Fundamental Properties of Floating-Gate Devices

Figure 1 depicts a p-channel floating-gate transistor. C_{tun} is a MOS capacitor to insure proper electron tunneling [8, 9]. The voltage at the floating-gate due to stored charge (V = Q/Cgate) is changed via a process called programming. The value of the

stored voltage is limited by electron tunneling due to a thin oxide barrier. In theory, the floating-gate voltage can be changed one electron at a time. Practical limitations create a floor of minimum charge added or removed from the gate. Readout of the floating-gate voltage is limited by the accuracy of utilized measurement instrumentation (and at the lowest limit by thermal noise of the floating-gate transistor).

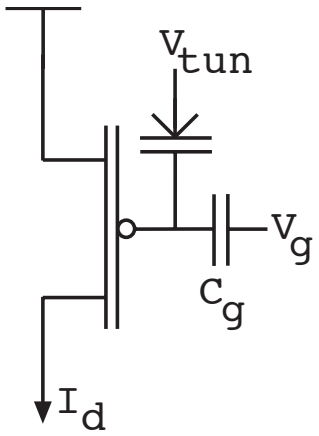


Figure 1: A transistor level schematic of the floating-gate pFET. The unlabeled capacitor is the tunneling capacitor, required for electron tunneling. The input capacitor allows input signals to be coupled into the floating gate terminal.

Circuits and systems using floating-gate transistors allow for expanded functionality compared to traditional designs. Many of the applications for floating-gate transistors include: being used to remove offsets in classical analog circuits, create permanent analog weight storage and reduce die area for large scale systems. Previous systems include an analog fourier processor, silicon cochlea, speech recognition system, and active pixel sensor imagers [10, 11, 12].

2.2 Modification of Charge in Floating-Gate Devices

The charge stored in the floating-gate is changed by three methods: UV radiation, Fowler-Nordheim electron tunneling, or hot-electron injection. We consider only the latter two cases as they can be implemented on-chip without need for additional hardware and controlled more precisely than UV radiation [8, 9]. Equation 1 is the

saturation drain current for a floating-gate transistor operating in sub-threshold (weak inversion), assuming a long channel device, $V_A = \infty$ and $V_{source} = V_{bulk}$.

$$I_d = I_o \exp \frac{\kappa(V_{fg} - V_T)}{U_T}. \tag{1}$$

In Eq. 1, U_T, is defined as the thermal voltage ($U_T = \frac{kT}{q}$), whose value at room temperature, 27° C, is approximately 25 mV. I_o is a device specific term. The floating-gate voltage is modeled as 2, where the nominal voltage terminals are the source, drain, well, tunnel node, and voltage at the gate input capacitor. The value of charge on the floating-gate, Q_{fg}, is changed via programming using the methods named previously.

$$V_{fg} = \sum_i (V_i * \frac{C_i}{C_{total}}) - \frac{Q_{fg}}{C_{total}} \tag{2}$$

Adjusting the charge on the floating-gate, will change effect gate voltage, V_{fg} as shown in Fig. 2. The floating-gate voltage is changed using Fowler-Nordheim electron tunneling and channel hot electron injection via impact ionization [8]. The shift of the curve may be presented as a shift of the threshold voltage for the device; or as a change of the saturation drain current for a fixed DC bias. For the p-channel floating-gate transistor, tunneling decreases the threshold voltage and decreases the drain current for a fixed bias. Hot-electron injection increases the threshold voltage and increases the drain current for a fixed bias. In effect, injection and tunneling are used as complimentary functions for programming a floating-gate transistor. Further explanation of our floating-gate programming methodology is presented in [8, 9].

2.3 Programming Multiple Floating-Gate Devices

Beyond programming a single floating-gate device, we need the capability to program multiple devices. The amount may range from a several devices to thousands of devices. The requirement for programming multiple floating-gate transistors creates a

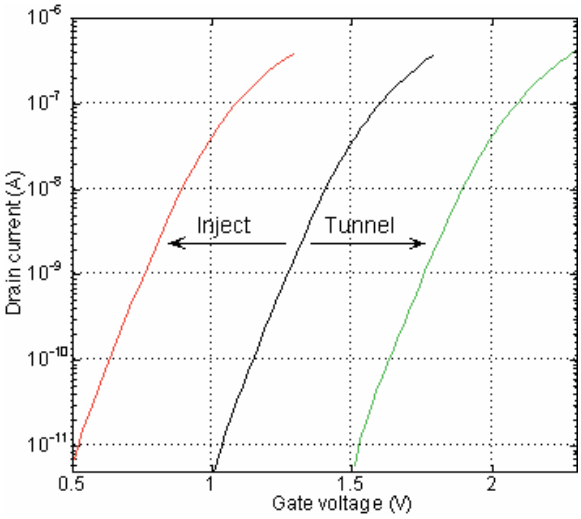


Figure 2: Gate sweeps of a pFET floating gate device [1] from a $0.5\mu m$ CMOS process. The effects of injection on V_T are seen as increasing its value. Electron tunneling decreases the value of V_T.

need for a programming architecture to facilitate efficient and accurate programming. Manually programming each floating-gate element without on-chip support circuitry would be an arduous task coupled with the notion that incorporation of offset removal with any complex system would become difficult. Manual programming, also requires user control over all aspects of electron tunneling and hot-electron injection. We propose to use automatic programming, which only requires a desired pFET drain current and selection of a floating-gate element to program, any other user control is not needed [5]. A floating-gate device can be programmed using an automatic programming algorithm, decreasing the complexity and time required to program such devices. An array based programming architecture is proposed which allows automatic programming of multiple floating-gate elements. This architecture contains most of the necessary components on-chip allowing for fast and timely programming.

2.3.1 Array Programming

The programming architecture utilizes an array based topology for the floating-gates elements to enable programming. A conceptual drawing of the topology is provided in

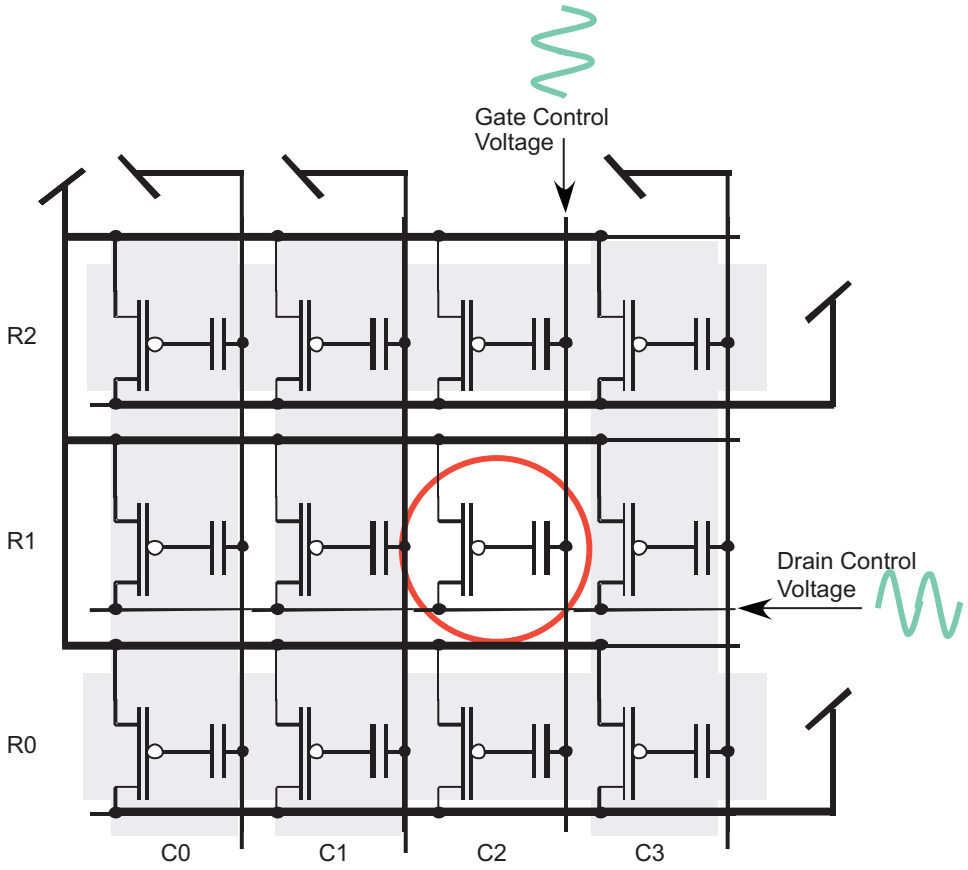


Figure 3: Schematic of the architecture used to program floating-gate elements. Shown is a single floating-element device in each cell [13, 14]; however, any arbitrary element can be used inside each cell of the array.

Figure 3. Each floating-gate element is an element of an MxN matrix/array. Selection of a specific floating-gate device is similar to writing a bit into an SRAM cell. For example, to select the floating-gate element found in column 4 and row 2: the gate input signal is applied to column 3, while the gates of all other columns are tied to V_{dd}. Since, the floating-gate transistors are pFET devices, they will effectively be shut off. For row selectivity, the drain and source for rows 1 and 3 are set to V_{dd}. This will deactivate all transistors in column 3 due to $V_{sd} = 0V$, except for the element located at row 2.

Having isolated a specific element, the next step is to program the device. For the array programming architecture, electron tunneling is used as a global erase function acting simultaneously on all floating-gate devices within an IC. Hot-electron injection is used to perform both coarse and fine addition of charge onto the floating-gate. Tunneling has been delegated as a global function because creating circuitry to enable selectivity for electron tunneling would require high-voltage switches located in each element of the matrix. This high voltage circuit would be very expensive in terms of die area, particularly if the number of floating-gate elements is large (on the order of 100 or greater). However, hot-electron injection is a process which allows selectivity with the same system as used to isolate a floating-gate element. Hot-electron injection has two requirements (from Chapter 3): a sub-threshold drain current and a large source-drain voltage, i.e. $V_{sd} > V_{dd,normal}$, for a $0.5\mu m$ process $V_{dd,normal} = 3.3V$. Upon closer inspection of the array programming system, we see that to isolate a single floating-gate element, all other elements have both their gate voltage set to V_{dd} and no drain current, with the exemption of elements located in the column of the selected device, which only have $V_{sd} = 0V$. Thus, the user has access to the gate, drain, and source of the isolated floating-gate element. Hot-electron injection may be performed on said element by creating a sub-threshold drain current (using the gate input) and applying a large V_{sd}. Given this array based element selection,

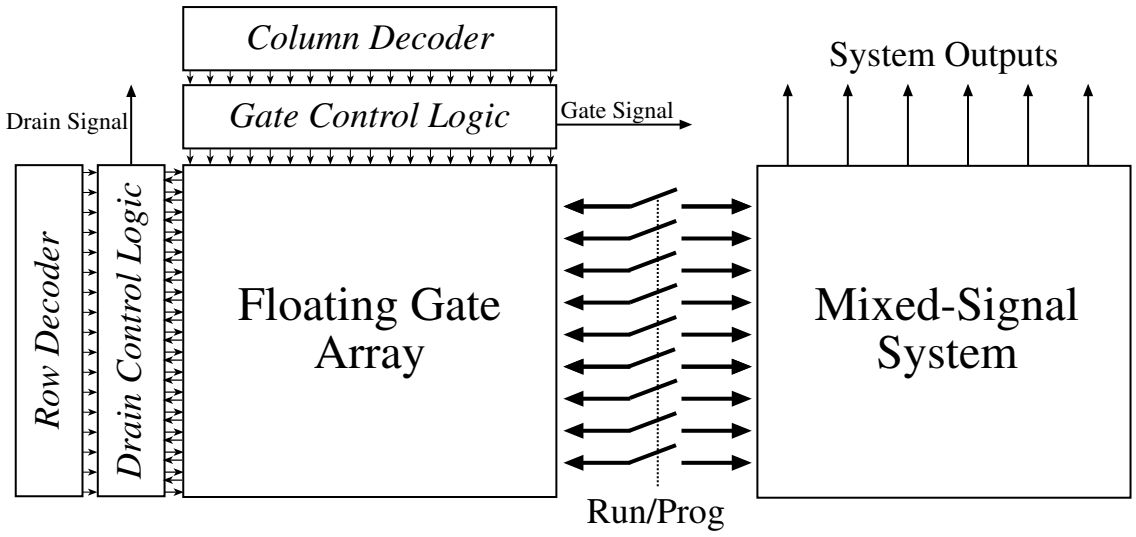


Figure 4: System-level block diagram used to implement the array based programming architecture. The decoders are used to control the logic circuitry. During *Run* mode, the floating-gate transistors are part of the mixed-signal system. During *Program* mode, the floating-gate transistors are separated from the system. Using the decoders, a single element is selected, and controlled via the Gate Signal and Drain Signal lines. Such control allows for programming of the device.

programming each floating-gate device in the array requires only selection of that device and applying hot-electron injection. In practice, however, electron tunneling is used to bring all elements in the array below a threshold current, in order to assure all devices are below the some minimum drain current (or below some maximum threshold voltage). The programming algorithm is applied to program each element of the array [15]. Thus, programming an entire matrix of floating-gate transistors using the array programming architecture requires only the desired drain currents/threshold voltages for each element; the programming system and algorithm will change each floating-gate element to the desired values efficiently without any further user input.

2.3.2 System Implementation

The array programming architecture is a modular design, enabling incorporation of the concept into almost any mixed-signal system. A system using the programming

architecture must also perform the function of making the array architecture transparent when not in use for altering the charge of the floating gate in each element of the array. This transparency requirement allows floating-gate transistors to exist as normal circuits that could be part of a given arbitrary system: e.g. multipliers, operational amplifiers, serial A/D convertors, etc. However, when the floating-gate array is required to be programmed, the array programming system will disable all elements of the IC except the floating-gate array and circuits required for the array programming architecture. This mode is designated the *Program* mode. Thus, in *Program* mode, only the array programming system is activated and operated as described in Section 2.3.1. Once programming of all elements is complete, the array programming architecture is disabled, and all floating-gate transistors are re-integrated with the mixed-signal system components; this mode is referred to as *Run* mode. A single clock signal, $Prog$, determines the mode of the programming system; a logic high state switches the system into the *Program* state, while a logic low sends the system into normal operation: the *Run* state.

Figure 4 details most of the components required to implement the array based programming architecture on chip for a mixed-signal system. A switching matrix is required to isolate the all the circuits of the mixed-signal system besides the floating-gate transistors. This switching matrix is controlled by the logic signal $Prog$. Column decoders are utilized to select a specific column for isolation. Outputs from the column decoder goto the *Gate Control Logic* which implements the gate isolation method described in the previous section. The input gate signal is routed to the selected column, while all other floating-gate transistor gates are tied to V_{dd}. The output from the row decoders goto the *Drain Control Logic* circuitry. This logic circuit implements the isolation via switching the source-to-drain voltage of each row. The selected row will have its V_{sd} voltage tunable, while other rows will have $V_{sd} = 0V$.

With the system components in place, programming an entire array of floating

gates is an automated procedure, using the programming algorithm. The system implementation allows direct control of each element in the matrix, along with its associated gate and drain voltages. The selected floating-gate transistor drain current is muxed out via the *Drain Control Logic* circuit. Tunneling is implemented as a global erase function, as stated in Section 2.3.1. After the programming of all elements is accomplished, the system is set into normal operation mode. The system circuitry requires the addition of a few I/O pins. The basic I/O pads are categorized into the decoder pins that are required for: selecting an element of the array, gate voltage pin, drain voltage, drain current pin (in the same pin) and a pad for the tunneling voltage.

2.4 Conclusion

We have described the basics of a floating-gate transistor in a CMOS process. We modify the charge on the floating-gate device using electron tunneling and hot-electron injection. In order to use multiple floating-gate transistors in a system, a basic programming infrastructure was designed, implemented, and tested. This is a key step for larger systems to floating-gate technology, in particular reconfigurable systems such as FPAAs.

CHAPTER III

SCALING OF FLOATING-GATE DEVICES IN DEEP SUB-MICRON CMOS PROCESSES

3.1 Introduction

Scaling of Floating-Gate (FG) devices is a key issue when working to improve the density of FG based memories, computing in memory systems, and Field Programmable Analog Arrays (FPAA). For example, how will an FPAA's operating frequency improve as the IC technology process is scaled down? Figure 5 shows a modeling summary of the capability in frequency of a particular FPAA device architecture as a function of process geometry used. Although the initial FPAA devices, built in 350nm process, have achieved frequencies in the 50-100MHz range, scaled FPAA devices should enable significantly higher frequencies, enabling RF type signals at 40nm and smaller IC processes [5]. Therefore, the potential of scaled down devices and the resulting computation, from a 350nm process down to a 40nm process, requires investigating both experimentally and analytically.

3.2 Basics of 40nm FG Devices

At 45nm / 40nm, compared to previous nodes (e.g. 90nm, 65nm), one sees a major change in the resulting MOSFET device. The gate insulator for MOSFET devices was changed from the time-tested SiO_2 to HfO_2 to reduce gate leakage in the thin insulator devices. Figure 7 shows a comparison of 350nm to 40nm FG devices. The higher ϵ of HfO_2 (25) enables a much thicker material while enabling increased coupling capacitance into the MOSFET surface potential (Ψ). The change in insulators enable a thicker insulator but with a smaller barrier potential (1.4eV versus 3.0eV), therefore

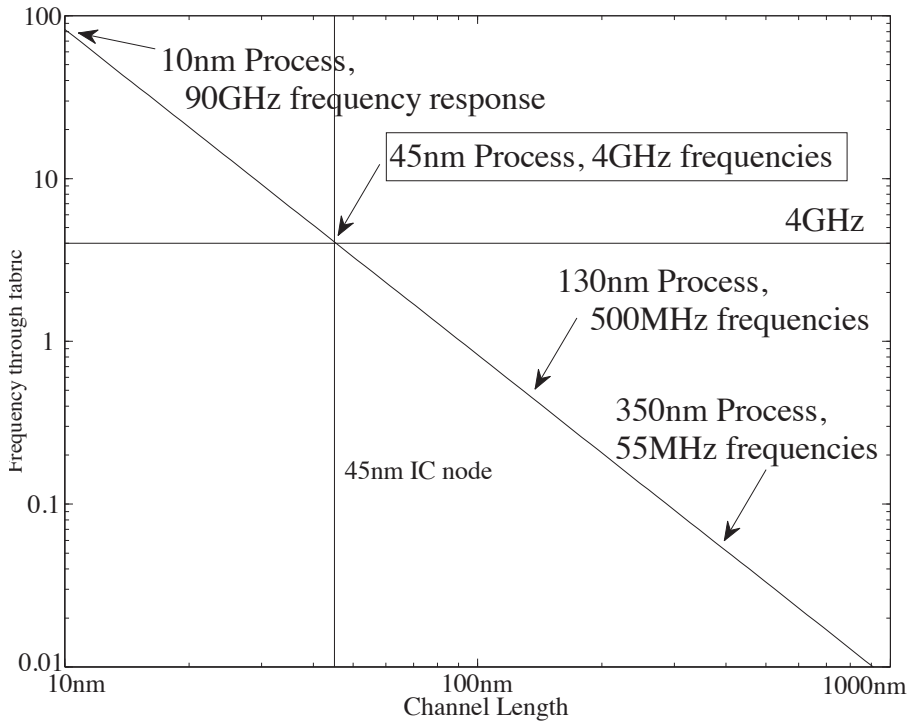


Figure 5: Frequency response of FPAA architectures as a function of minimum channel length.

for a square barrier we would expect lower leakage current than a comparable 350nm device. From experimentally built FG devices in a 40nm IC process, we can measure the channel current for gate sweeps and drain sweeps, as shown in Fig. 7. Using the measured drain current from an FG gate sweep, starting at the pFET subthreshold region to near threshold region and ending in the above threshold region; we are able to extrapolate an effective κ of 0.373 and a threshold current of 100nA. From the measured drain current versus swept drain voltage we will be able to extract the resulting $g_s r_0$ of these devices, which includes the effect of overlap capacitances.

To determine viability of floating-gate devices in this process node, we first ask whether these new FG devices hold charge, (at least sufficiently long for testing our systems). Furthermore, do we see behavior of sufficiently long hold-times to expect reasonable 10 year lifetime results, similar to EEPROM devices. Experimental measurements to date have shown FG devices that hold charge over days with negligible

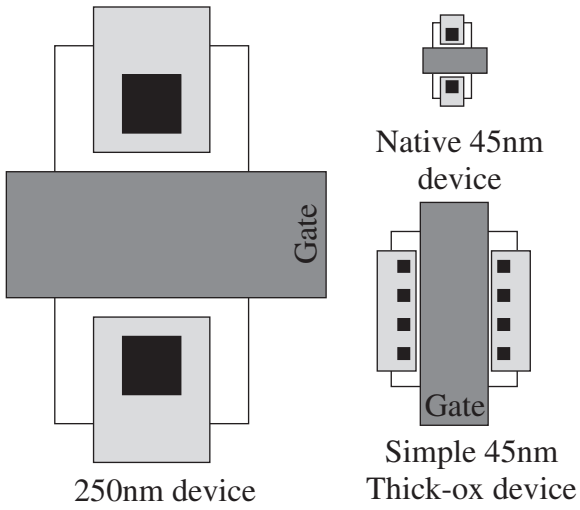


Figure 6: Multiple FG devices layouts (250nm and 40nm processes). We show a typical 250nm device, a typical 45nm device, as well as a thicker insulator 45nm device. The source-drain to substrate / well capacitance is significantly less in the 45nm approach, a key parameter for making dense arrays of floating-gate devices.

change in the stored charge. To provide an explanation our of assertion, we look at the square barriers between the 350nm and 40nm devices in Fig. 7. The change in insulators enable a thicker insulator; but, with a smaller barrier potential (1.4eV versus 3.0eV). Therefore given a square barrier we would expect lower leakage than a 350nm device. Electron tunneling current depends on the exponential of a term proportional to the thickness and proportional to the square-root of barrier energy ($E_{barrier}$); the classic expression for tunneling through a square barrier

$$I_{tun} = I_{tun0} \exp\left(-\frac{2\sqrt{2m^*}}{\hbar}\sqrt{E_{barrier}}t_1\right), \tag{3}$$

where t_1 is the insulator thickness, m^* is the effective mass of an electron, and I_{tun0} is an experimentally determined constant for the particular insulator [16].

The FG devices created and tested were made using the thick oxide device available in the 40nm process. This thick oxide is made from HfO_2, and can be thought of as similar to a 250nm standard CMOS device. We can, also, assume using the thicker oxide device will enable long (i.e. 10 year) charge storage lifetimes. Figure 6 shows

scaled pictures of different transistor sizes. The thicker insulator device for 45nm process, despite having an effective gate thickness similar to a 250nm process, has a drain-source parasitic capacitance lower than a standard 250nm process. Minimizing these parasitics is critical for frequency performance for any implementation, with an added benefit of higher density of devices due to smaller size. Decreasing the entire FG device size to a typical 45nm device with a thicker insulator, typical of EEPROM type devices, should be possible but is not experimentally tested.

3.3 40nm Floating-Gate Device Measurements

Figure 8 shows the experimental methods and related measurements for electron tunneling through the HfO_2 gate insulator in a 40nm FG thick oxide device. For both 350nm and 40nm FG devices, electron tunneling behavior is described by Fowler-Nordheim tunneling effects and equation. As an aside, 250nm FG devices would also follow such behavior. The modified 40nm FG FET insulator results in higher electron tunneling current because of the smaller barrier to Si (1.4eV) versus the classic SiO_2 to Si barrier (3.0eV). Fowler-Nordheim tunneling, or tunneling through a triangle barrier, models electron tunneling current as

$$I_{tun} = I_{tun0} \exp\left(-\frac{4\sqrt{2m^*}}{3\hbar}\frac{E_{barrier}^{3/2}}{q\mathcal{E}},\right) \tag{4}$$

$$I_{tun} = I_{tun0} \exp\left(-\frac{2\sqrt{2m^*}}{\hbar}\sqrt{E_{barrier}}t_1\frac{2E_{barrier}}{3qV_{ox}}\right) \tag{5}$$

where q is the charge of an electron, and $\mathcal{E}$ is the electric field in the insulator [17]. Equation 5 is derived from Eqn. 4 via substitution of $\mathcal{E}$ for V_{ox}.

When a high potential is applied across the tunnel oxide, we assume the FG device is still acting similar to a normal device. Next, we group terms in the electron tunneling Eqn. (5) and simply the equation to:

$$I_{tun} = I_{tun0} \exp\left(\frac{V_o}{V_{tun} - V_{fg}}\right) \tag{6}$$

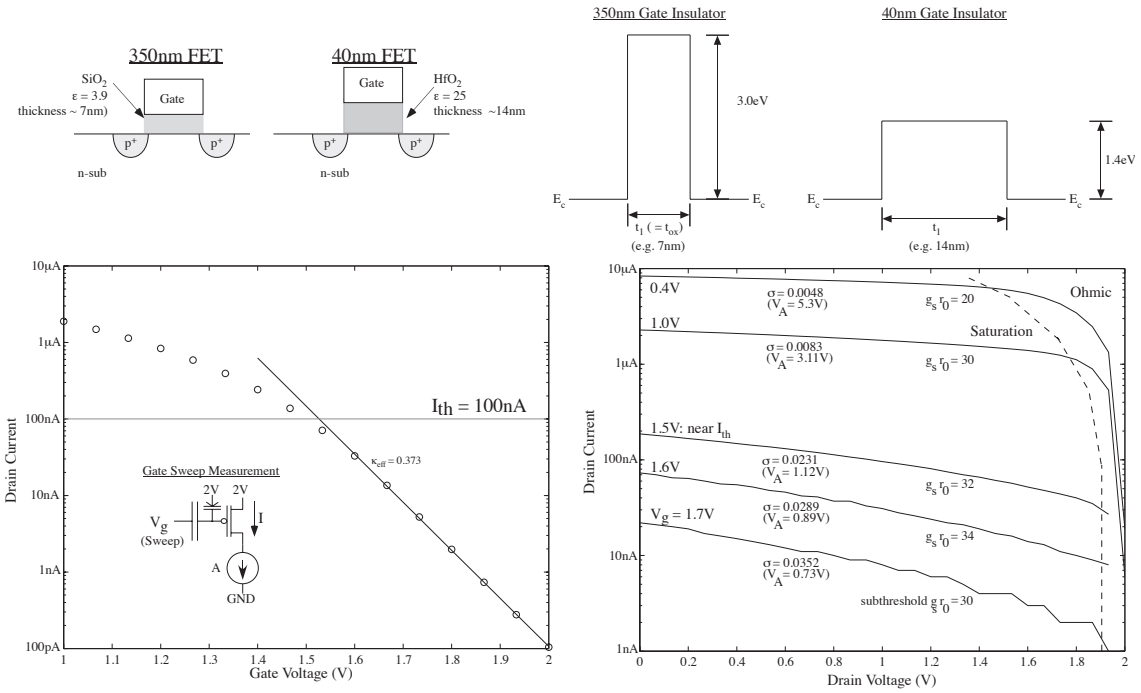


Figure 7: Illustration of the comparison of a 350nmFG FET and a 40nm FG FET. We compare the typical device used for a 350nm FET device versus a thicker insulator available 40nm FET device that could enable long-term lifetimes for FG devices. Basic gate and drain sweep curves are presented for the 40nm thick oxide floating-gate device.

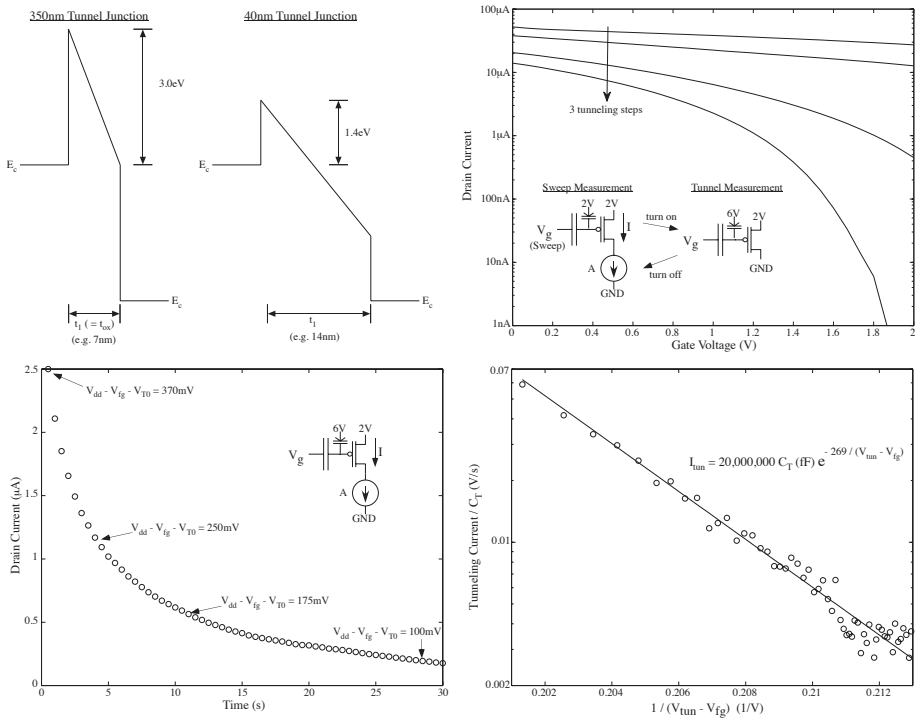


Figure 8: Measured drain current from a single 40nm FG device demonstrating electron tunneling between sweeps. Comparison between 350nm and 40nm processes for electron tunneling are rooted in looking at the resulting band-diagrams. One can take several gate sweep curves with tunneling between the curves. Tunneling occurred at 6V supplied to V_{tun}, with delays on the order of a minute between curve sweeps. Curve sweeps were taken with V_{tun} at 2.0V. We can measure the time course of tunneling. From the resulting current (above-threshold) current measurements, we can extract floating-gate voltage (V_{dd} - V_{fg} - V_{T0}), enabling characterizing tunneling current versus tunneling terminal voltages (V_{tun} - V_{fg}). We regressed tunneling current per unit total floating-gate capacitance (C_T) versus 1 / (V_{tun} - V_{fg}) enabling a direct comparison of the data with the theoretical expression for Fowler-Nordheim tunneling. We also plot a curve fit to that theoretical expression in (6).

where V_o is a parameter including the other exponential terms in (5). We relate the terminal voltages to transistor elements as $V_{tun} - V_{fg} = V_{tun} - V_{dd} + V_{T0} + (V_{dd} - V_{fg} - V_{T0})$.

For our above-threshold current measurement versus time, we take our model of current-voltage relationship (verified by data to be reasonable) as

$$I = \frac{K\kappa}{2}\left(V_{dd} - V_{fg} - V_{T0}\right)^2 \tag{7}$$

$$I = \frac{\kappa^2 I_{th}}{4U_T^2}\left(V_{dd} - V_{fg} - V_{T0}\right)^2$$

$$V_{dd} - V_{fg} - V_{T0} = \frac{2U_T}{\kappa}\sqrt{\frac{I}{I_{th}}} \tag{8}$$

where threshold current, I_{th}, is $2KU_T^2/\kappa$, and we extracted I_{th} as 100nA from our data on this particular device. From these measurements of V_{fg}, we can extract the tunneling current by writing KCL at the floating-gate as

$$C_T\frac{dV_{fg}}{dt} = I_{tun} \tag{9}$$

The resulting formulation allows us to take a numerical derivative to see the resulting tunneling current, enabling the plot in Fig. 8 and resulting curve fit of Eqn. (6). We see from Fig. 8, the tunneling behavior of a thick oxide FG device in a 40nm process fits accordingly with Fowler-Nordheim tunneling and comparable to a 250nm process.

With electron tunneling established for 40nm devices, we will focus on hot-electron injection next. The lower energy barrier between HfO_2 impacts channel hot-electron injection by reducing the barrier for electrons injecting into the insulator. For SiO_2 barriers, a wide range of the effects were limited by hole impact ionization and we expect in these processes that the correlation will be far stronger. We expect that we will need similar voltages for injection across processes.

Figure 9 shows the experiments performed to measure hot-electron injection in the 40nm FG devices. We take a drain current versus gate voltage sweep to determine the initial starting condition of the device (similar to electron tunneling measurements).

An initial tunneling step is performed to bring the device into an "off" position, i.e. very low drain currents. Next, we took a gate voltage sweep to determine an initial starting curve, however we produced a sharp jump in current. This was caused by a major ammeter range change at 2μA, which caused the drain voltage to drop for a short time to a voltage below GND, enabling enough electric field on this FG device to inject, as seen by the immediate step in current resulting from an decreased level of FG charge. To mitigate this effect, we turned off auto ranging on the ammeter and performed the initial gate sweep again. The second curve in Fig. 9 does not have an inflection point.

Next, the drain voltage was pulsed to large V_{ds} values, similar to the method described in Chapter 2. We pulsed the drain voltage and measured the resulting drain current, I_d, until the perceived current value stopped changing appreciably.

Figure 10 shows the measured change in drain current for a fixed pulse width, T, versus starting the drain current. We show these measurements for three values of — V_{ds} — , and from this result, we extract the resulting slope at a fixed current (i.e. 10nA). This slope is $1/V_{inj}$; V_{inj} = 96.7mV. This value is utilized in floating-gate programming algorithms when an accurate desired V_{fg} or I_d is required (e.g. precise bias currents/voltages) [18].

Often in programming algorithms, we make use of an effective linear difference equation(s) for early steps in hitting analog targets [18]. Figure 11 shows the linearized difference equations of resulting drain current after an injection event versus initial current.

3.4 Floating-Gate Devices in 130nm

Similar to the 40nm process, we made test structures of floating-gate transistors in a 130nm CMOS process. The gate insulator is made from SiO_2 unlike the 40nm and below processes. To obtain the best charge retention, we made the floating-gate

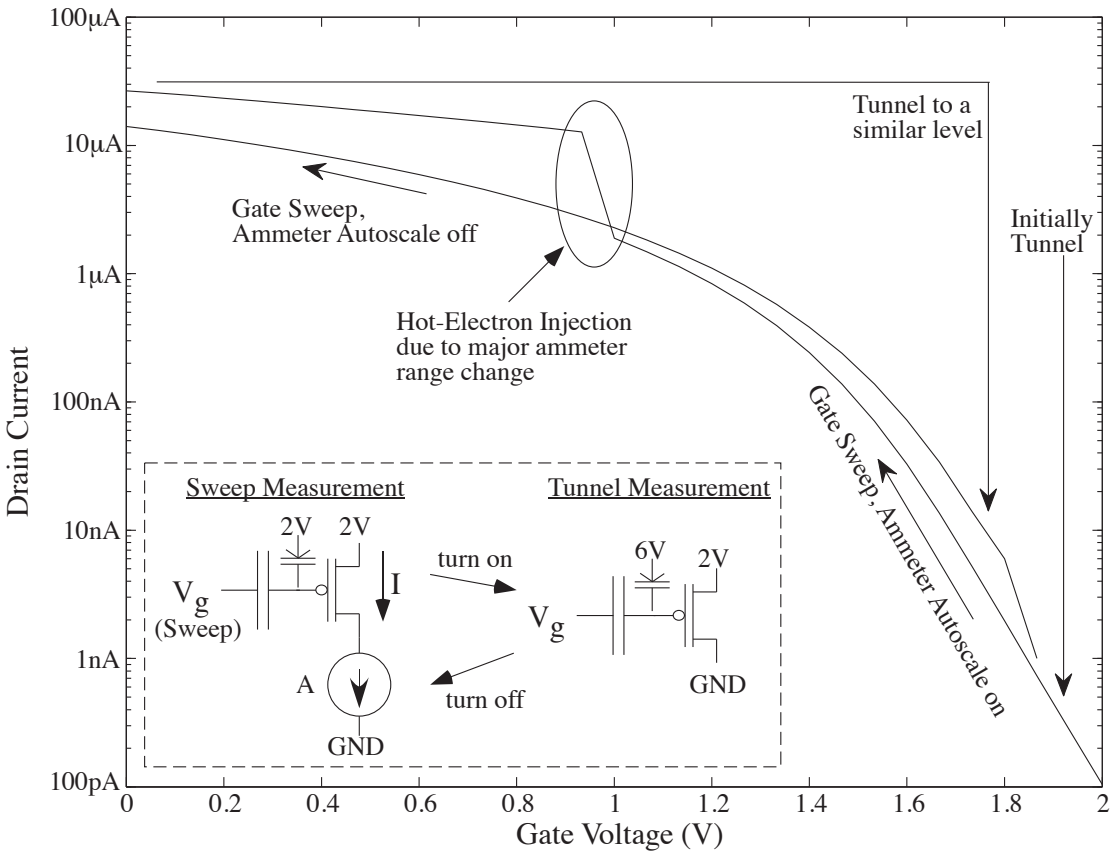


Figure 9: Hot-electron injection experimental setup. Ammeter auto ranging was turned off due to unintended injection occurring.

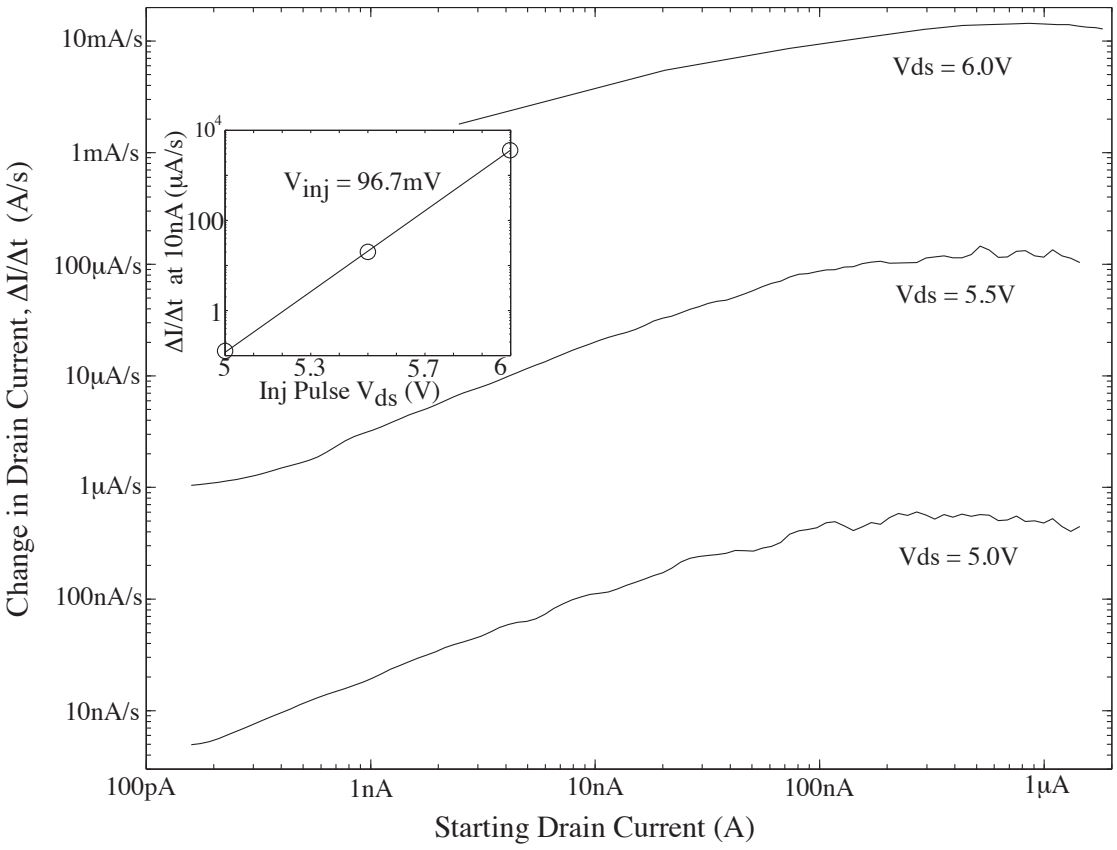


Figure 10: Pulsed injection measurement results for various V_{ds} values.

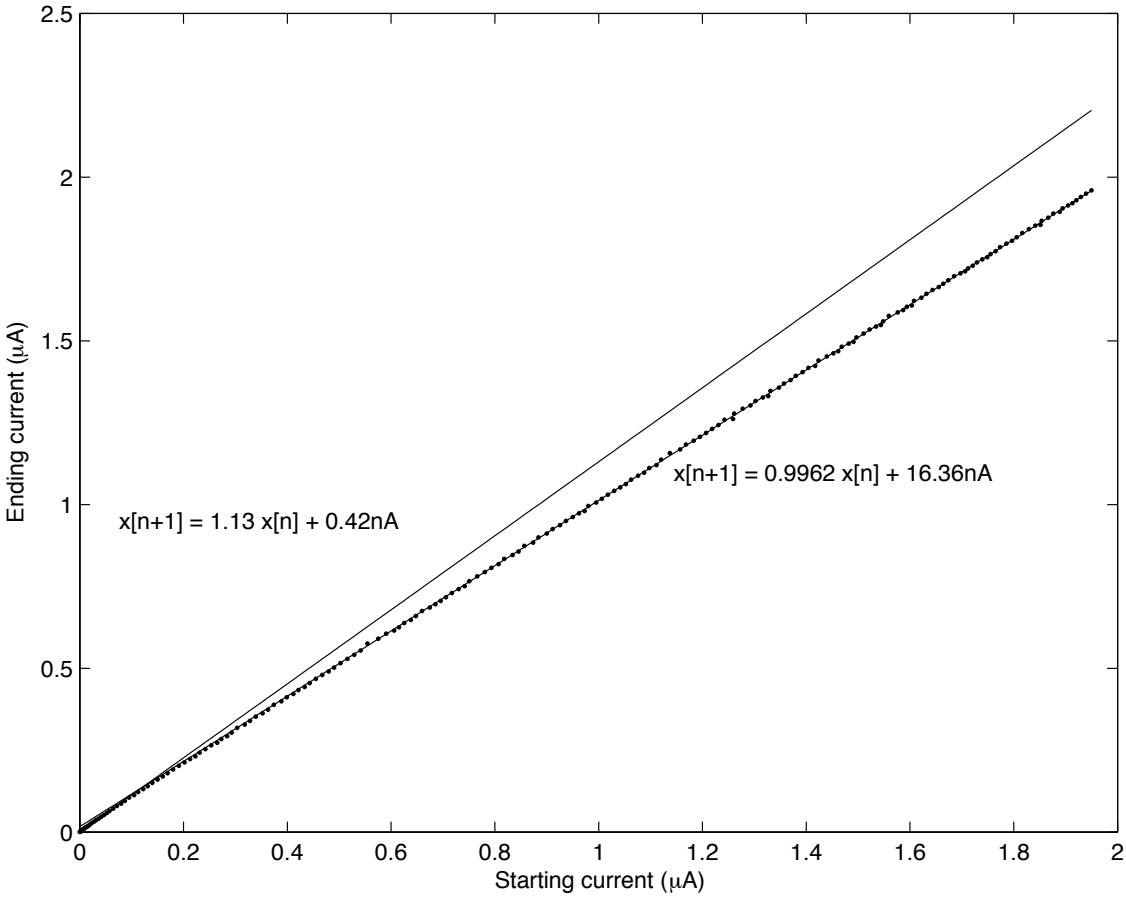


Figure 11: Linear difference equations used to determine ending drain current via hot-electron injection.

device using the thick oxide option. This transistor is similar in characteristic to a 250nm node (again, similar to the 40nm process). Figures 12 and 13 show the results of electron tunneling and injection from a 130nm floating-gate device using the thick oxide of the process. The measurements were performed in the same manner as described above for the 40nm devices. We are able to show comparable tunneling and injection characteristics between the 40nm and 130nm devices.

3.5 Conclusion

We have shown the scaling of floating-gate CMOS devices across 130nm and 40nm technology. The transistors were made using the inherent process thick oxide devices. We were able to show both tunneling and hot-electron injection behavior comparable to the 350nm transistors. We expect charge retention numbers to be comparable between the process nodes, although further experimentation is needed. The gains in density from scaling can be further amplified by utilizing the process thin oxide source/drain junction geometries along with thick oxide gates. Such a device can lead to lower junction capacitance and higher layout density.

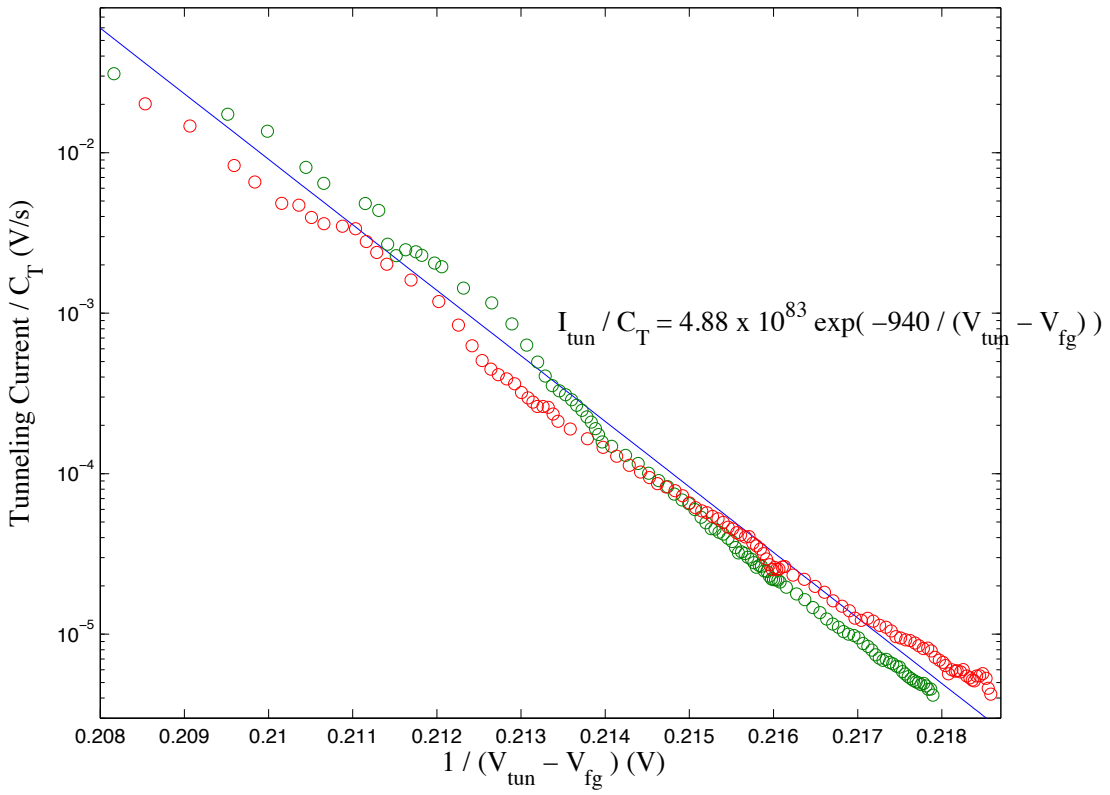


Figure 12: Fowler-Nordheim plot from a 130nm floating-gate device.

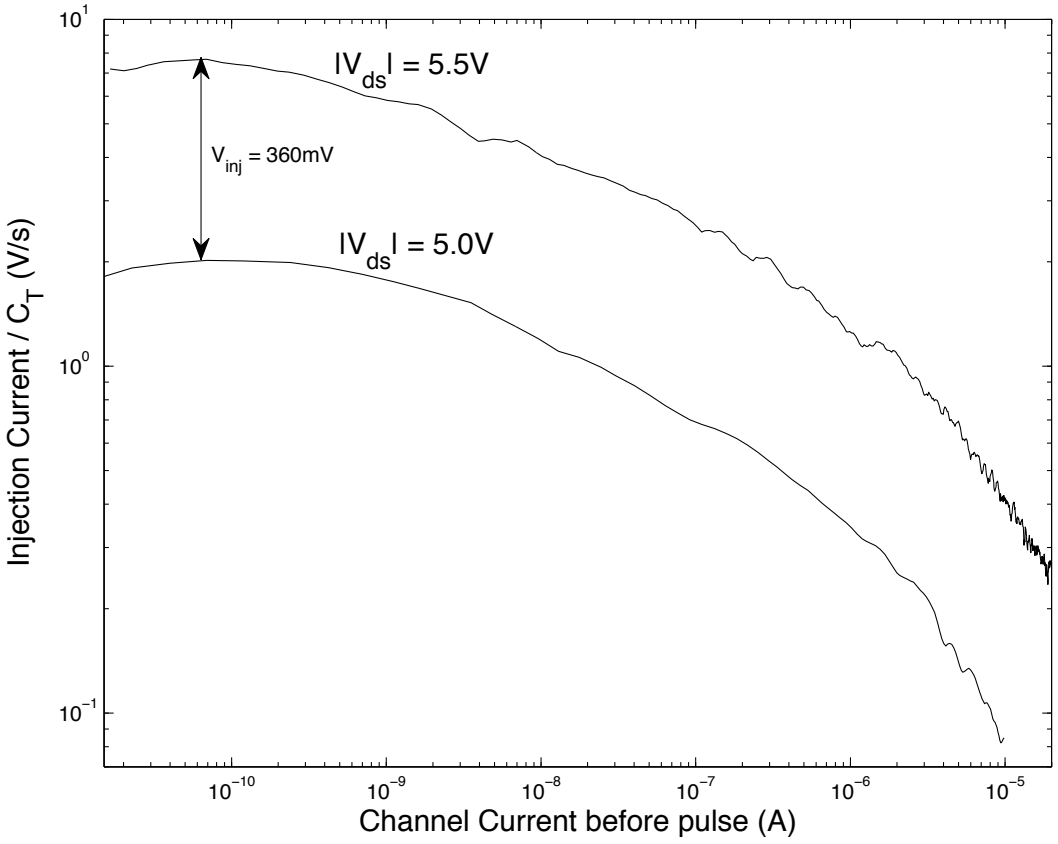


Figure 13: Hot-electron injection data from a 130nm floating-gate device.

CHAPTER IV

ANALOG CIRCUITS USING FLOATING-GATE DEVICES

Floating-gate transistors can be efficiently used as analog biases and non volatile analog weight storage [10]. More importantly, floating-gate transistors can be used in various analog circuits to help mitigate defects due to the fabrication of integrated circuits [19]. Device geometry mismatch between multiple transistors due to imperfections in fabrication manifest themselves as a deviation in the threshold voltage of a transistor. We will show the mitigation of this mismatch term by inserting floating-gate transistors in analog circuits.

4.1 Model for Offset Removal

One method to characterize device mismatch is the measurement of variation in threshold voltage. As discussed in Chapter 2, floating-gate transistors enable direct manipulation of the effective transistor threshold voltage. The offset voltage is the difference of threshold voltage between a reference device and "matched" devices. This offset voltage can be reduced, if all reference and matching transistors are designed as floating-gate devices.

To model offset removal between two devices, we start with Eqn. 2 and define the summation term as V'_{fg}

$$V_{fg} = V'_{fg} - \frac{Q_{fg}}{C_{total}} \tag{10}$$

Inserting Eqn. 10 into Eqn. 1 yields

$$I_d = I_o \exp \frac{\kappa(V'_{fg} - \frac{Q_{fg}}{C_{total}} - V_T)}{U_T}. \tag{11}$$

The effective threshold voltage from Eqn. 11 is

$$V_{T,eff} = \frac{Q_{fg}}{C_{total}} + V_T \tag{12}$$

The process of matching one reference floating-gate device with another (or more) floating-gate devices requires the ability to match $V_{T,eff}$ of each device. From Eqn. 12, changing the stored charge on the floating-gate enables the devices to be matched and reduces their offset mismatch to within the resolution of the measurement device utilized. This methodology will be used to reduce threshold voltage mismatch in CMOS transistors and used to created matched circuits. Mismatch due to gate area (W/L) is neglected because V_T differences are the larger cause of errors in VLSI systems, and increasing the matching of gate dimensions may be done during layout.

4.2 Offset Removal in Differential Amplifiers

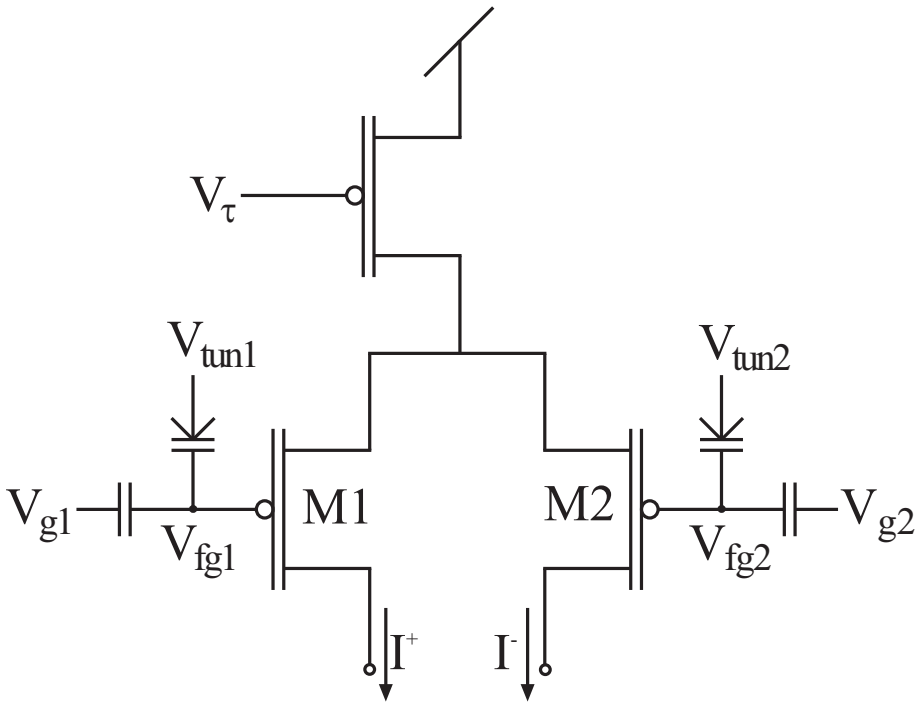


Figure 14: Transistor level schematic of the floating-gate differential pair.

Offset removal in a different analog circuits will be shown. All measured data is

obtained from a $0.5\mu m$ CMOS chip. Figure 14 shows a differential pair with floating gates at each of the input terminals. With the addition of floating gates, we have the ability to remove the input-referred offset voltage ($V_{in,offset}$) caused by a mismatch in threshold voltage between both transistors. $V_{in,offset}$ is related to $V_{T,eff}$ of the floating-gate devices by the following equation

$$V_{in,offset} = V_{T,eff1} - V_{T,eff2}. \tag{13}$$

Altering $V_{T,eff1}$ of M1 and/or $V_{T,eff2}$ of M2, can be used to reduce $V_{in,offset}$ to 0. Figure 15 displays the measured results of reducing $V_{in,offset}$ in a differential pair over several programming iterations. Due to the unique nature of the circuit, $V_{in,offset}$ may be changed to any arbitrary value allowing the system to have calibrated differential pairs with specific defined offsets. This ability to create offsets at specific locations enables the differential pair for use in comparators with user-defined trip points. Measured results of a differential pair programmed to arbitrary offset values are shown in Fig. 16.

Analogous to removing the input referred offset voltage of a differential pair, floating-gate transistors are utilized to remove threshold mismatch effects found in current mirrors. The threshold voltage mismatch causes a non-unity input/output current ratio (assuming the mirror is designed for a ratio of 1). Figure 17 depicts a schematic of the floating gate based current mirror. The floating-gate transistors in the current mirror were programmed to have equivalent $V_{T,eff}$, thus reducing offset mismatch in the current mirror. Figure 18 shows the measured current mirror data after offset reduction.

Combining a floating-gate differential pair and floating-gate current mirror, a standard 9-transistor wide-output operational transconductance amplifier (OTA) has been built [20]. The circuit has three sources of mismatch errors: the input differential pair,

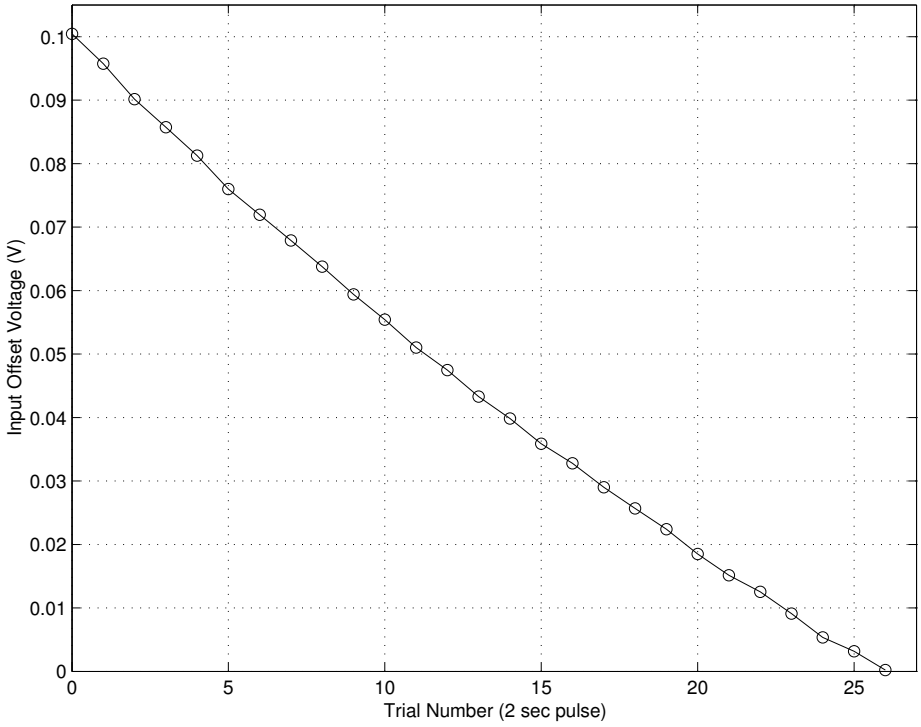


Figure 15: $V_{in,offset}$ is progressively trimmed over successive iterations. The final measured offset voltage is $< 1mV$, limited by instrument resolution

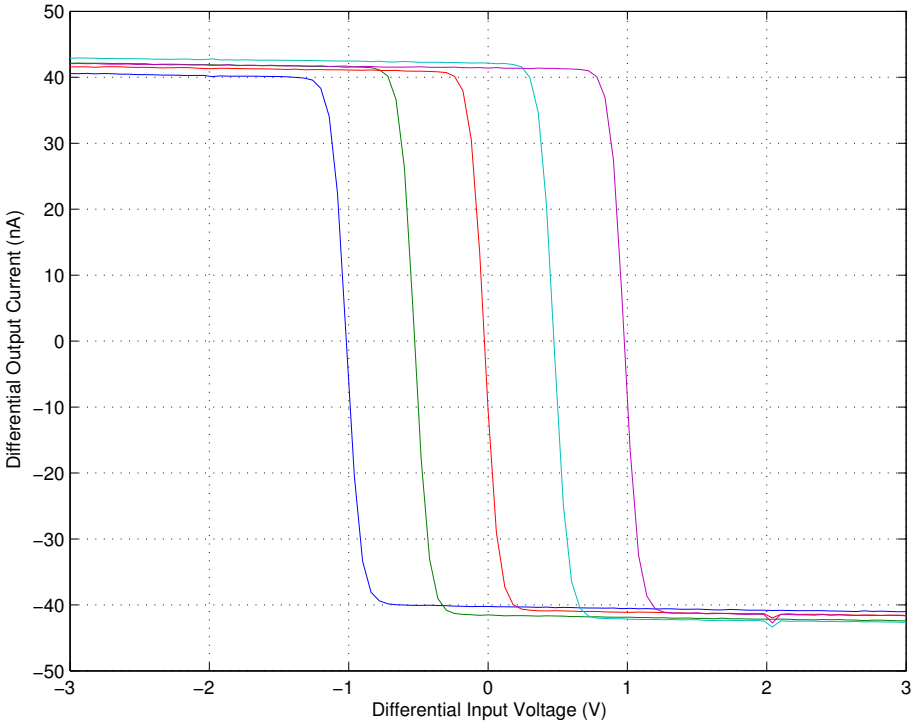


Figure 16: User-defined values for $V_{in,offset}$ using array programming. A differential pair was taken and its offset voltage changed in increments of 0.5 V from -1.5 V to 1.5 V using the offset creation program.

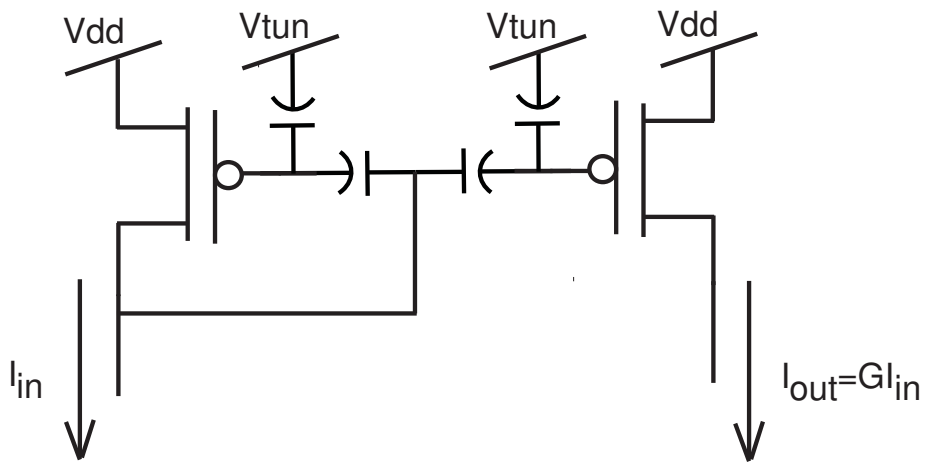


Figure 17: Schematic of a current mirror using floating-gates. G is the multiplication/gain factor of the current mirror.

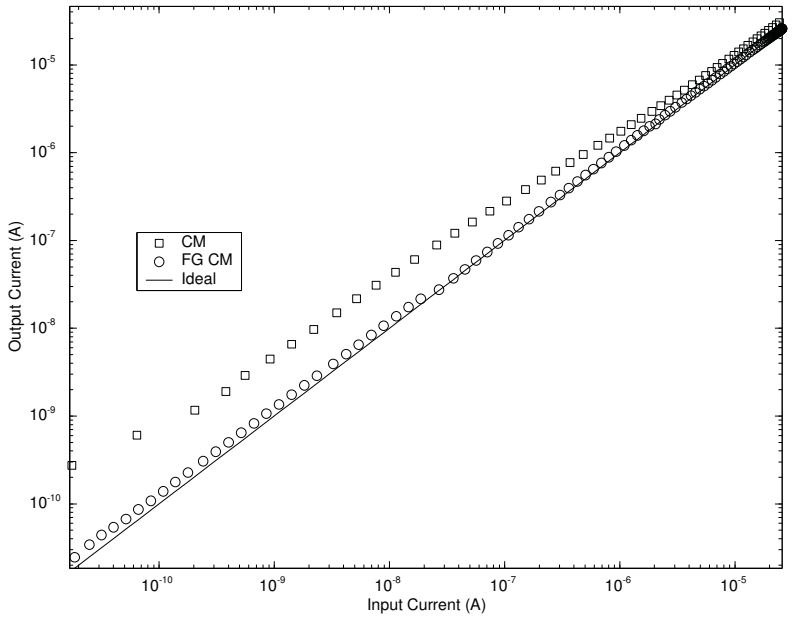


Figure 18: Floating-gate based current mirror after offset reduction.

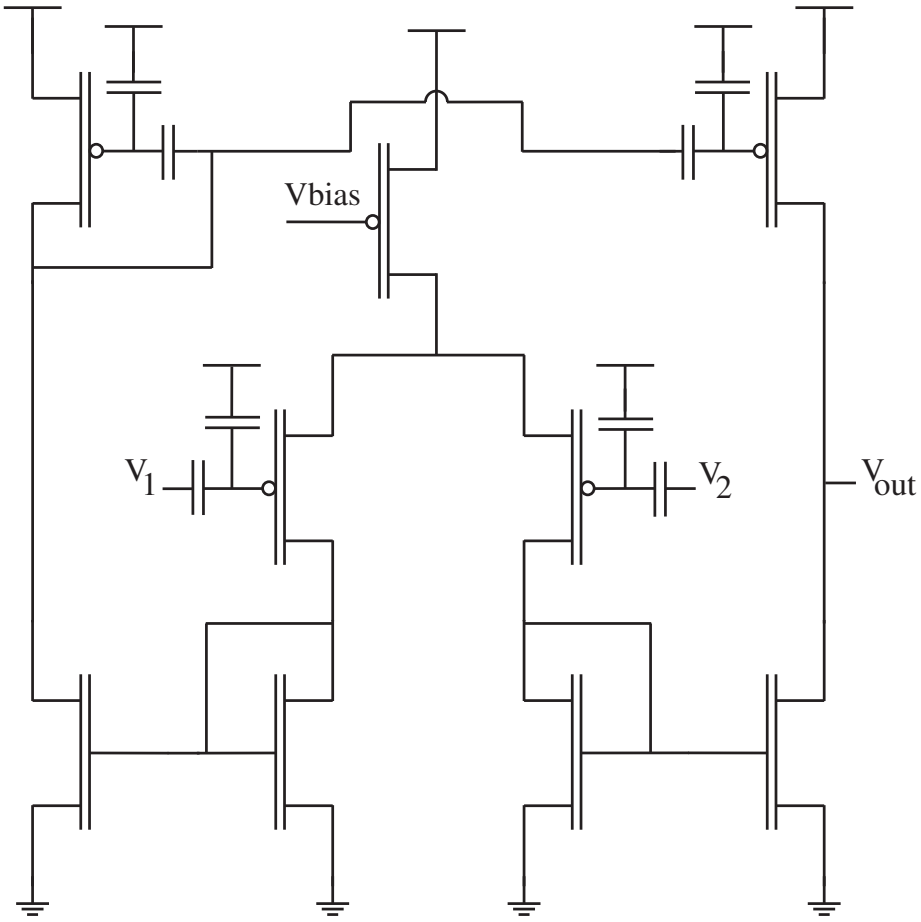


Figure 19: Schematic of the floating-gate operational transconductance amplifier.

the pMOS current mirror, and two nMOS current mirrors. The W/L for all transistors is 1. All pMOS circuits are utilizing floating-gate transistors to perform offset removal.

For the OTA circuit, offset removal follows techniques similar to that presented earlier, with a minor variation. The offset mismatch of the NMOS current mirror can not be reduced directly, due to the lack of NMOS floating-gates in this process. The PMOS current mirrors are programmed to remove the total mismatch of the NMOS and PMOS current mirror combination. The amplifier is operated in sub-threshold with a DC bias current of 20 nA. The resulting gain, A_v, is approximately 40 V/V. The measured output $V_{out,offset}$ is 5 mV. The input referred offset voltage, $V_{in,offset}$, is $125\mu V$. The limitation of offset reduction is due to instrumentation resolution during

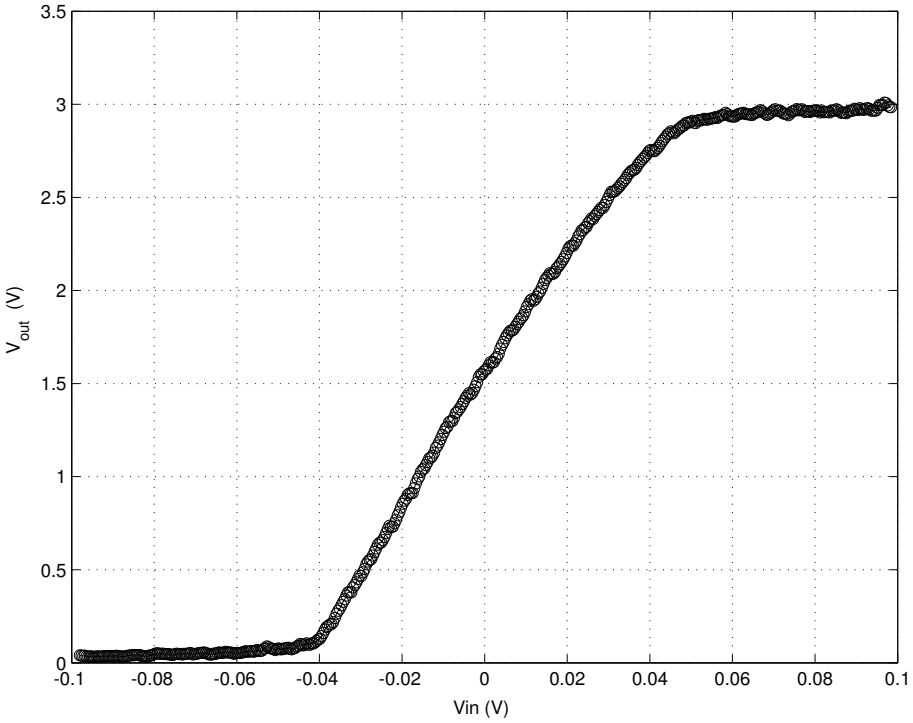


Figure 20: Measured transfer function of the floating-gate OTA.

measurement.

Parameter	**Value**
A_v	40 V/V
$V_{out,offset}$	5 mV
$V_{in,offset}$	0.125 uV

Table 1: Summary of Experimental Results for the 0.5μm floating gate based OTA

4.3 Offset Removal in Gilbert Multipliers

Using the floating-gate differential pair, a four-quadrant multiplier, based on the Gilbert multiplier, has been built. Figure 21 shows the schematic of the floating-gate multiplier, essentially, two differential pairs are utilized. As before, the array programming architecture is incorporated for automatic and efficient programming. The $V_{T,eff}$ of each transistor are programmed to be equal.

Results from an offset removed Gilbert multiplier are shown in Fig. 22. An IC has been fabricated on the MOSIS 0.5μm, from which the results have been taken. A

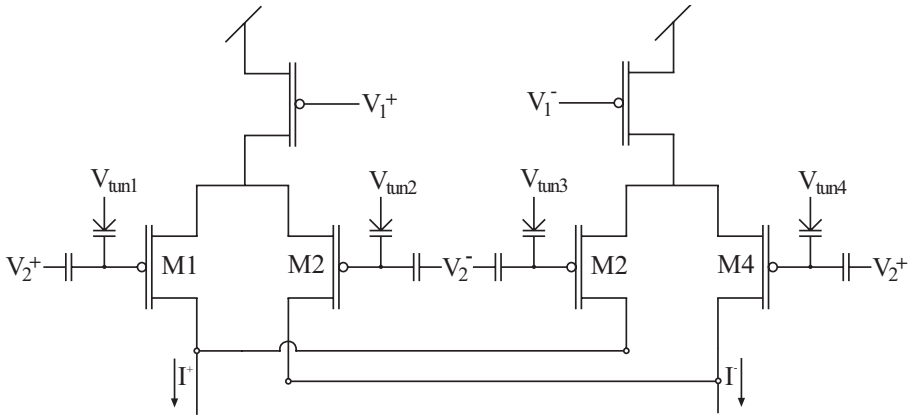


Figure 21: Transistor level schematic of a floating-gate multiplier circuit. The multiplier is based on the Gilbert Multiplier.

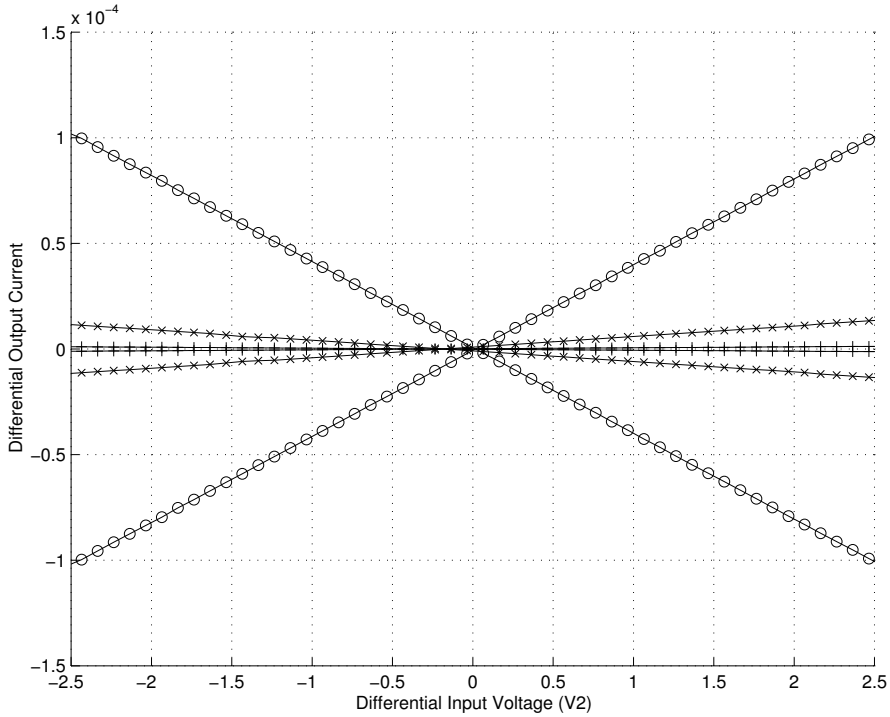


Figure 22: Multiplier results after offset removal. I-V curves for various values of the differential voltage V1. Each floating-gate differential pair was tuned such that the difference between I^+ and I^- was zero. The offset for each curve is approximately zero.

similar offset removal method from above was used to tune the multiplier. The differential voltage V_1 is kept constant for each sweep of V_2 to obtain Fig. 22. After offset removal, the multiplier input referred offset voltage is measured to be approximately 1 mV. Changing the multiplication, by modifying the value of V_1 has no effect on the value of V_{offset}, as was expected. Multiplication of the differential signals V_1 and V_2 are clearly seen and an intercept through the origin. The linearity of the I-V curve is due to a small value for κ_{eff} for all transistors, similar to that reported as above. A Gilbert multiplier created with floating-gate transistors having κ_{eff} in the higher range, between 0.5 and 0.7, would also show values of V_{offset} comparable to the low κ_{eff} multiplier presented.

4.4 System Examples

Low-level analog circuits, such as operational transconductance amplifier (OTA), are important pieces in building data converters and most mixed-signal systems. A reconfigurable system should enable the ability to change the attributes and properties of the designed application, e.g. change filter corner frequencies. Previously, we discussed basic analog sub-circuits and OTAs for use in reconfigurable systems. We will use the floating-gate (FG) OTA from above as the basis for programmable $G_m - C$ filters. Taking the OTA from Fig. 19, it is modified for full differential operation. Common-mode feedback (CMFB) compensation is performed using a floating-gate based feedback structure [21]. The CMFB circuit is built using two capacitors to sum the two outputs; thus, computing the output common mode, and directly applying this signal as feedback to the output current sources. Measured results from a $0.5\mu m$ CMOS process are shown in Fig. 23.

Biquad filters are a common filter design used in analog and mixed signal systems. Using the FG OTA, we have designed low-pass and band-pass biquad filters with programmable frequency responses. Figure 24 is a low-pass biquad filter implementation. The filter corner frequencies were programmed to different values ranging from 200 kHz to 2 MHz. The Q of the filter was also programmed to various values. The FG CMFB common-mode rejection was measured at -50 to -60 dB. These values agreed with simulation.

Band-pass biquad FG OTA filters were also designed. Figure 25 shows band-pass filter results with tuning ranges of 25 kHz to 100 kHz for the low corner. The high corner has programming tuning ranges of 1 MHz to 4 MHz. Similar to the low-pass filter, the Q of the band-pass filter is tunable via programming.

4.5 Conclusion

We have shown the ability to reduce the effect of threshold mismatch in analog circuits, such as differential pairs, current mirrors, OTAs, and gilbert multipliers. The reduction of input referred offset voltage (and other offset voltages) will enable higher performance and accuracy analog circuits. Also, we have used floating-gates as tuning knobs for various analog systems (i.e. filters). Floating-gate devices are a simple solution to the problem of multiple non-volatile bias sources. They are compact and provide sufficient range for the systems wherein they are utilized.

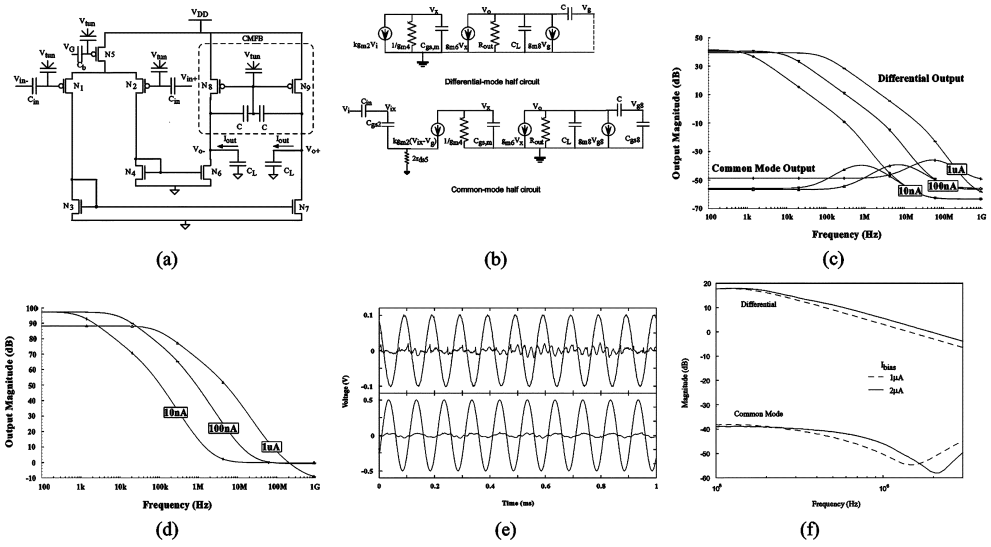


Figure 23: Fully differential FG-OTA with FG CMFB circuit and measurements. (a) Circuit schematic. (b) Small-signal circuit schematics. (c) SPICE simulation results of small signal circuit for various bias currents. (d) SPICE simulation results of CMRR versus frequency. (e) Transient common-mode response. Response is shown for 10-kHz input common-mode signal at 200 mV_{pp} and 1 V_{pp}. (f) Experimental frequency response for two different programmed bias currents.

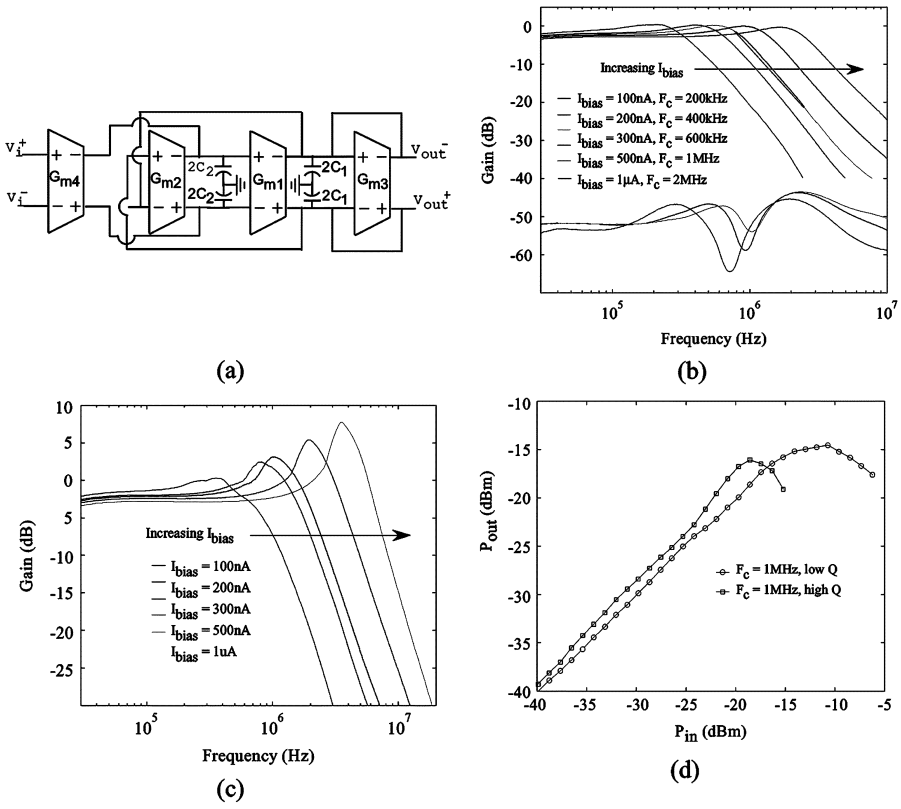


Figure 24: Programmable low-pass filter biquad and measurements. (a) Block diagram for the programmable low-pass filter biquad using FG-OTAs. (b) Measured differential and common-mode gain. (c) Measured differential gain showing the Q variations. (d) Measured plot to compute the 1-dB compression point for a LPF tuned at 1 MHz for two different programmed Q values.

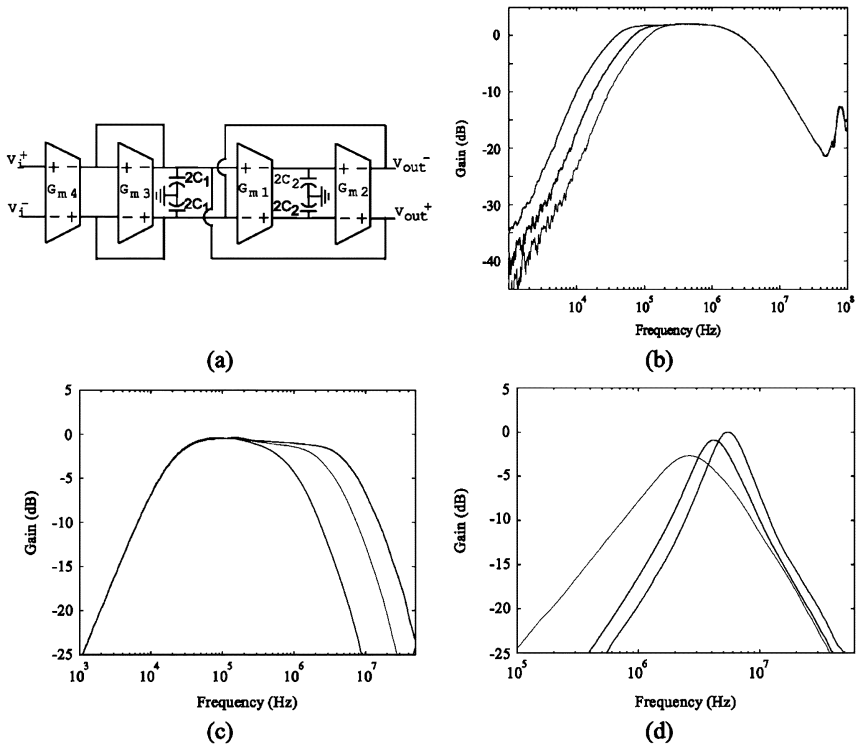


Figure 25: Programmable bandpass filter biquad and measurements. (a) Block diagram for the programmable bandpass filter biquad using FG-OTAs. (b) Experimental results showing the programming of the low corner of the Bandpass filter. c) Experimental results showing the programming of the high corner of the bandpass filter. (d) Experimental results showing programming of the low corner of the bandpass filter for different Q values.

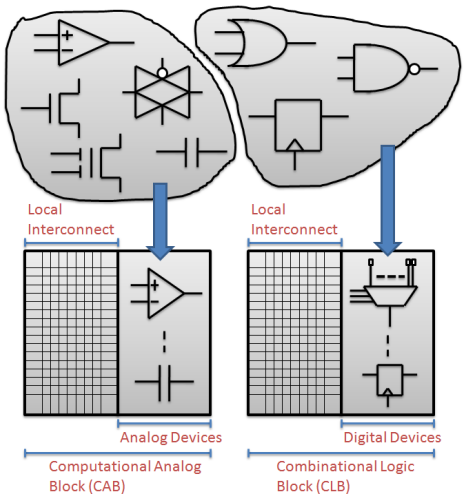
CHAPTER V

RECONFIGURABLE TILE-ARRAY MIXED-SIGNAL PLATFORM

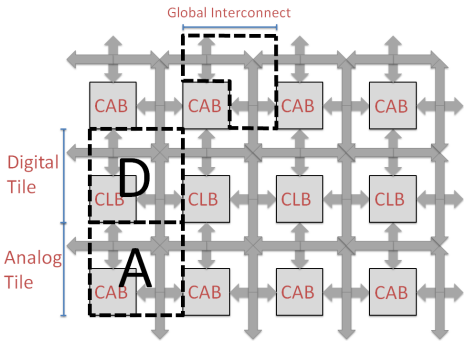
The utilization of reconfigurable FPGA systems in the digital design arena has exponentially increased over the last decade. Systems with ASIC designs are becoming less prevalent while FPGA-based system designs are increasing. The benefits of rapid prototyping, quicker time to market, and in-field reprogrammability are important benchmarks which enabled the rise of the FPGA [22, 23, 24]. In the analog/mixed-signal design domain, the uptake for reconfigurable chips has not seen the same rate. The benefits of using reconfigurable mixed-signal chips are equivalent to the digital domain: faster design cycle, reduced time to mark, and in-field reconfigurability. A mixed-signal FPGA or Field-Programmable Array of Analog and Digital Devices (FPAADD) has been developed to capitalize on the benefits of mixed-signal reconfigurability [25]. The generality and flexibility of the FPAADD enable it to implement a vastly larger application space over previous reconfigurable systems. Examples of application systems include, but are not limited too: data converters, digitally assisted analog computation, industrial control, machine learning, mixed-signal processing, digitally tunable analog circuits, to biologically inspired neuromorphic circuits.

5.1 Mixed-Signal Architecture

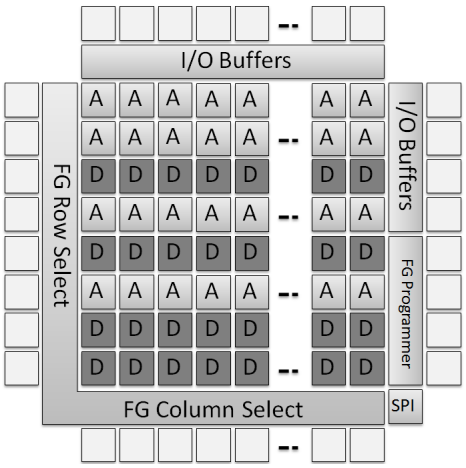
The computational blocks are clusters of computational elements and an interconnect network called the local interconnect. Analog components are clustered together to form the Computational Analog Blocks (CABs) while digital components are clustered into Combinational Logic Blocks (CLBs).



(a)



(b)



(c)

Figure 26: The general architecture of the FPAADD: a) Left, analog devices (MOSFETs, capacitors, etc.) are grouped together with local interconnect, a sea of reconfigurable switches for connecting the devices together, to form Computational Analog Blocks (CAB). Right, digital devices (Flip-Flops and look-up tables) are grouped together with local interconnect to make Combinational Logic Blocks (CLB). b) Interchangeable digital and analog tiles are built from either a CLB or a CAB with reconfigurable routing that allows signals to propagate between tiles (global interconnect). c) System view of the FPAADD at the top level.

In the FPAADD, the choice of analog devices range in complexity from discrete transistors and capacitors to FG input operational transconductance amplifiers (FG-OTAs). Other devices include: transmission gates, and multiple-input translinear elements (MITEs) [26]. The choice of digital devices are LUTs and flip flops.

CLBs and CABs are arranged in a tile-able Manhattan style global interconnection scheme (Figure 26b). An analog tile comprises a CAB, two connection blocks (C-Blocks), and a switch block (S-Blocks). The C-Blocks allow inputs and outputs from the CAB to connect to the global routing tracks, while the S-Block routes nets on global tracks through the chip. The digital tile is the exact same but with a CLB. The two different tiles are completely pin compatible.

FG transistors are used for the switches and state storing elements on the chip. The dynamic range of the FG switches allow for ON performance comparable to transmission gates with parasitic capacitance of a single FET, with leakage currents an order of magnitude less than standard SRAM based alternatives [27]. The non-volatile nature of the floating-gates means the chip does not have to be reprogrammed on power up. The continuum between the on and off states allow the routing infrastructure to perform tasks other than just connecting nets: tunable delays, current biases, and vector matrix multipliers (VMM), for instance, are all easily implementable by the interconnect.

The core of the FPAADD is an array of these tiles. The tiles are interleaved on a row by row basis with a higher density of digital rows on the bottom and analog rows on the top. The rest of the chip is floating-gate selection and programming infrastructure (controlled by an SPI bus), and buffered and non-buffered I/O. The top level arrangement of the chip is shown in Figure 26c.

The Manhattan-style routing architecture chosen for the FPAADD is the parameterizable one as understood by the VTR software. Things like the number of global tracks, track lengths, number of inputs and outputs from cells, etc. are all variables.

In general, arbitrarily cranking up these variables usually leads to an increase in routing options. This can increase the chance that the place and route heuristics successfully find a routing solution to any given target circuit, and or make impossible to route circuits routable. The biggest trade off in doing so, is that an increase in routing options comes from an increase in the number of switches on any given net, and thus increases the parasitic delay of any routed signal, the dynamic power consumed on transitions, increases static power, and reduces the fraction of the silicon devoted to the actual computational elements.

The FPAADD routing architecture will be presented parameterizable, and with the specific values of any variable. The choices for said variables was, to a certain extent, a bit arbitrary. Though certain performance goals, i.e.. a minimum desired routed device to device bandwidth did set upper bounds on the number of allowable connections on certain nets.

5.1.1 Floating-Gate Switch

The most basic and ubiquitous component of any highly reconfigurable architecture is the switch and the switch's state storage. In the majority of modern FPGAs, this is implemented by a single nFET whose gate is driven by SRAM. The FPAADD, instead, uses floating-gate transistors as the switch and memory.

Building up the local interconnect and high level portions of the chip is greatly facilitated by defining some circuit symbols: Figure 28a shows the symbol used for a floating-gate pFET switch, and Figure 28b the symbol for when a floating-gate is used as the gate input to a larger circuit like an inverter.

An open circle, as shown in Figure 28c, denotes when a switch is used to connect two abutting net lines. When a switch is used to allow connectivity between two crossing net lines an open circle is drawn over the crossing of the two nets (Figure 28d). Figure 28e shows the symbol for an s-switch connection topology, an open

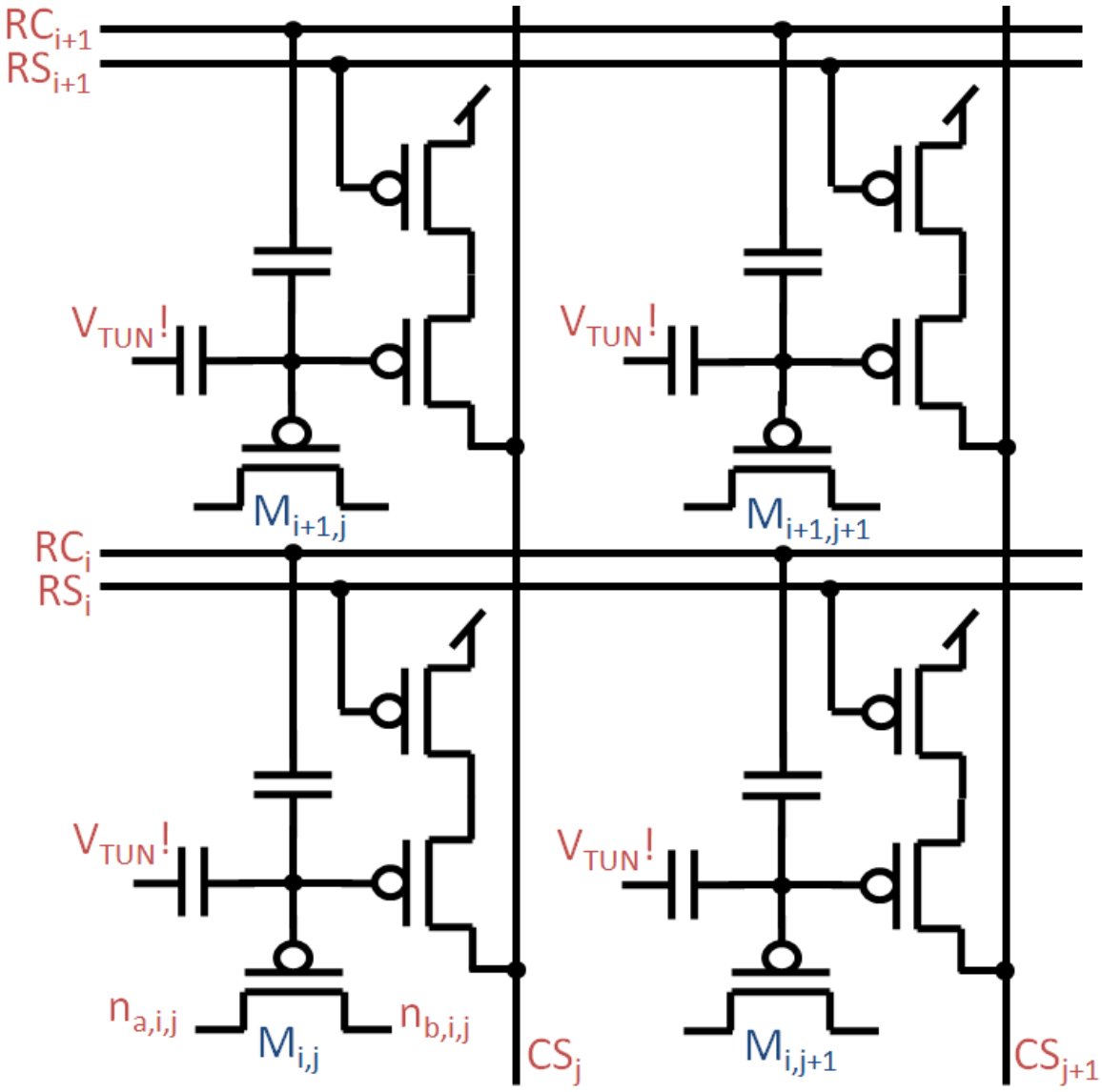


Figure 27: Programming is achieved by globally removing charge from the floating-gate nodes through C_{TUN} via Fowler-Nordheim tunneling, and then selectively adding charge through $M_{i,j}$ with impact carrier hot channel electron injection. Injection of charge per row is controlled by the selection lines CS_i, and per column by the drain lines CS_j.

square. The s-switch allows a signal entering from any side to propagate across, make a turn, split in two directions, allows two nets to cross each other, or turn away from each other.

5.1.2 Combinational Logic Block

The Basic Logic Element (BLE) is the building block of the digital circuits. The standard BLE is a k-input look-up table whose output is either registered or not by a flip-flop. Shown in Figure 29 is the BLE implementation used in the FPAADD. Instead of using a standard flip-flop, a JK-FF is used that can be configured as a T-FF or a D-FF. The clock can be routed from the local interconnect, the BLE's look-up-table, or come from a global signal. These choices were made to allow of high density synthesis of asynchronous counters.

Figure 30 shows that the CLB is comprised of NO number of BLEs and a sea of local interconnect. The inputs to each BLE come from either any of the NI primary inputs to the CLB or from the outputs of any BLE in the CLB. The NO outputs of the CLB are hardwired to the outputs of the BLEs in a one-to-one fashion. The configuration of the local interconnect allows for a deterministic and guaranteed routing solution for any clustering of any NI inputs and NO BLEs. Where $NO = 4$ and $NI = 8$.

5.1.3 Computational Analog Block

The CAB is the analog equivalent of the CLB. It is a cluster of analog devices and local interconnect, however, instead of a homogeneous set of devices, the CAB in the FPAADD contains: floating-gate based operational transconductance amplifiers (OTAs), switched capacitor optimized transmission gates, MOSFETs (either common centroid pFETs or nFETs), capacitors, and multiple-input translinear elements (MITEs: floating-gate pFETs with multiple input control gates). This set of devices was chosen to make the FPAADD CABs compatible with the generic CABs from the

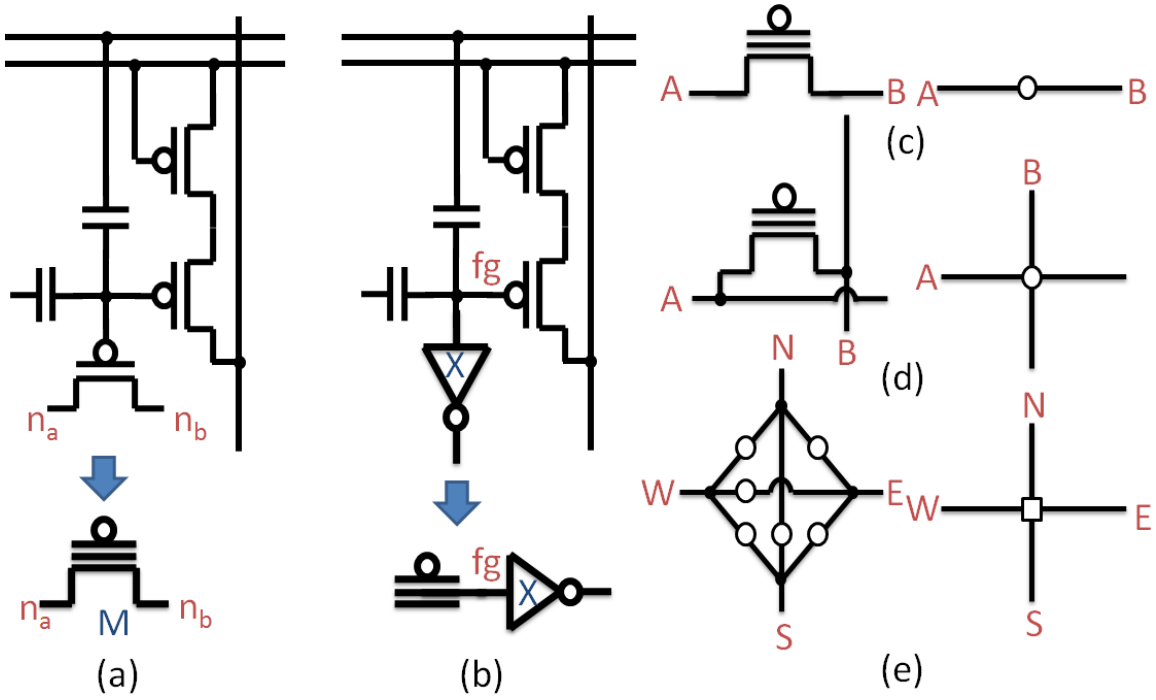


Figure 28: a) pFET switch with floating-gate memory and circuit symbol, b) Circuit symbol for a floating-gate memory element setting the gate input voltage of an inverter, c) a pFET floating-gate switch connecting two abutting nets, d) a pFET floating-gate switch connecting two crossing nets, e) six pFET floating-gate switches implementing an s-switch.

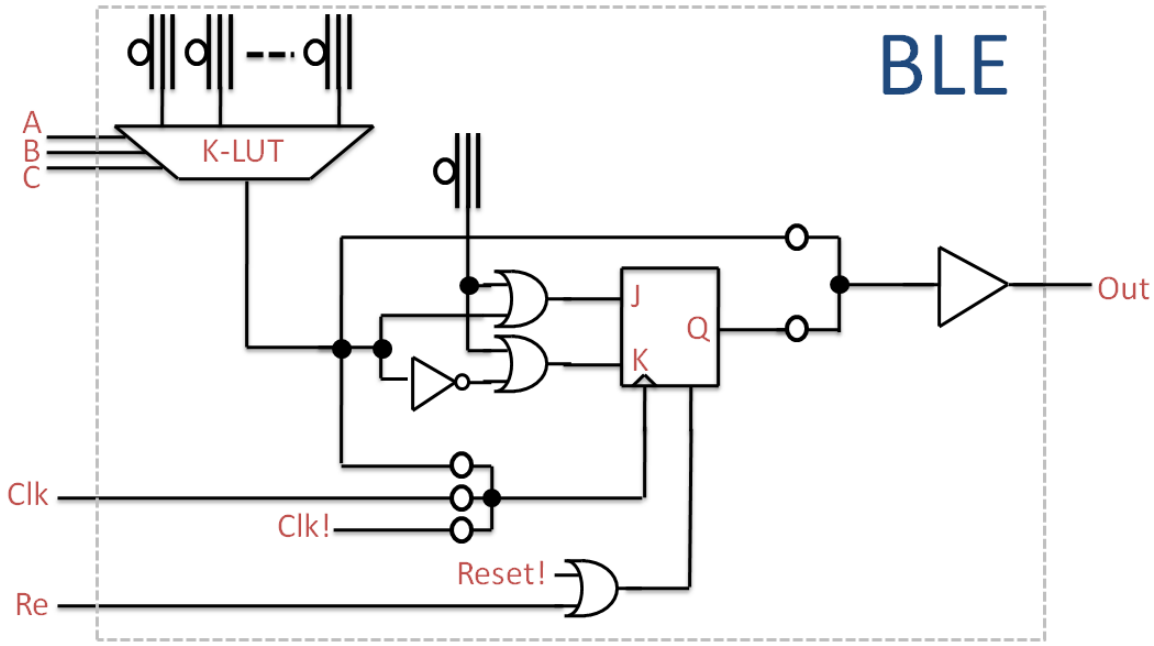


Figure 29: The BLE is a 3-input LUT whose output can be registered with a FF. The register is implemented as a JK-FF. It cab be configured as a standard FF or a T-FF, with the clock originating from the local interconnect, the output of the LUT, or a global line.

previous FPAA of [5]. Inputs to the devices come from the NI primary inputs, the two hardwired V_{DD} and *gnd* signals, or the outputs of any device in the CAB. The NO outputs of the CAB are multiplexed from the set of CAB device outputs. This was chosen because the number of devices in the CAB exceeded that in the BLE and it was desired to keep the same number of I/O in the CAB as in the CLB: $NO = 4$ and $NI = 8$.

While the routability of the CLB was complete, this is not the case for the CAB. The existence a completely deterministic and guaranteed routing solution for all combinations of NI inputs, NO outputs and CAB devices depends on whether the clustering can be partitioned such that the implied input/output relationship of the devices is preserved: an output can go to multiple inputs, but multiple outputs can not go to a single input. In the CLB, the inputs and outputs are well defined, as is the case with CMOS digital gates. While an OTA may have well defined inputs and outputs,

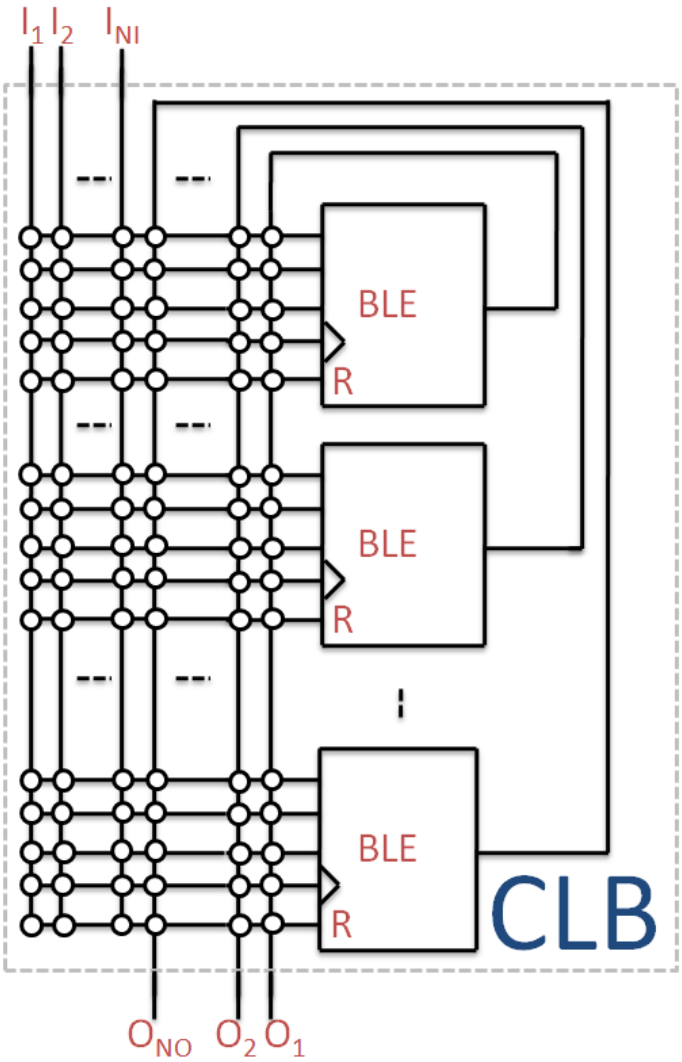
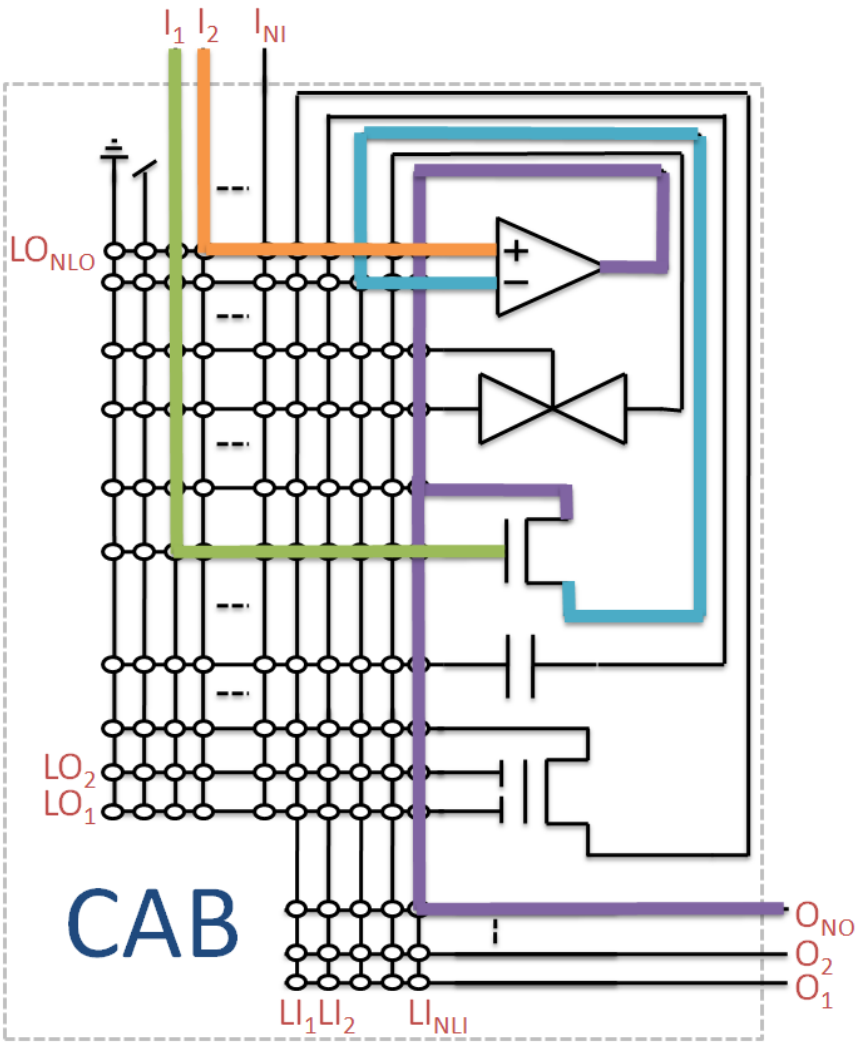


Figure 30: The CLB comprises multiple BLE devices and a sea of local interconnect. The outputs from the NO number of BLEs are the primary outputs from the CLB, and the inputs to the BLEs come from the NI number of primary CLB inputs and the NO BLE outputs. $NO = 4$ and $NI = 8$ for the FPAADD.



CAB architecture showing devices and local interconnect. Inputs to the local interconnect are vertical lines and outputs from the local interconnect are horizontal. I and LI are the primary inputs to the CAB and the outputs from the CAB devices respectively. O and LO are the primary outputs from the CAB and inputs to the CAB devices respectively. The example wiring shows a configured logarithmic amplifier circuit.

and the gate of a MOSFET is easily classified as an input, classifying the sources or drains of a MOSFET, for instance, as either inputs or outputs is rather arbitrary. If partitioning of the circuit to be clustered in the CAB preserves these mappings then the cluster is guaranteed to route in a deterministic manner. Since many analog circuits do not partition this way, this does not automatically mean they will not route, the output multiplexor allows for limited support of shorting of outputs. If outputs are to be shorted, and if the output is also a primary output, then the output multiplexor can handle this. The only time output shorting will fail is if two devices are to short their outputs, and this net does not propagate out of the CAB or to the input of any device in the CAB, and all CAB output lines are already occupied with other nets.

5.2 Manhattan Routing Design in FPAA

5.2.1 Global Interconnect

The global interconnect follows a standard track based scheme with C-Blocks getting inputs and outputs out of the CABs and CLBs and onto the tracks. S-Blocks allow track segments to be connected across, or to make turns. Figure 31 shows a two by two array of tiles where each tile contains either a CAB or CLB and global interconnect: two C-Blocks and an S-Block. There are 11 tracks in the north-south direction, and 11 in the east-west direction.

The C-Blocks in the FPAADD are implemented as a completely populated floating-gate matrix. The C-Blocks are not fractional, all inputs and outputs from the computational blocks have access to every track, and all track segments span one tile length.

The S-Blocks are a diagonal arrangement of s-switches (one buffered, ten passive) that allow signals to propagate across or to change directions into neighboring tiles, but the diagonal nature keeps the signals on the same track number as they started.

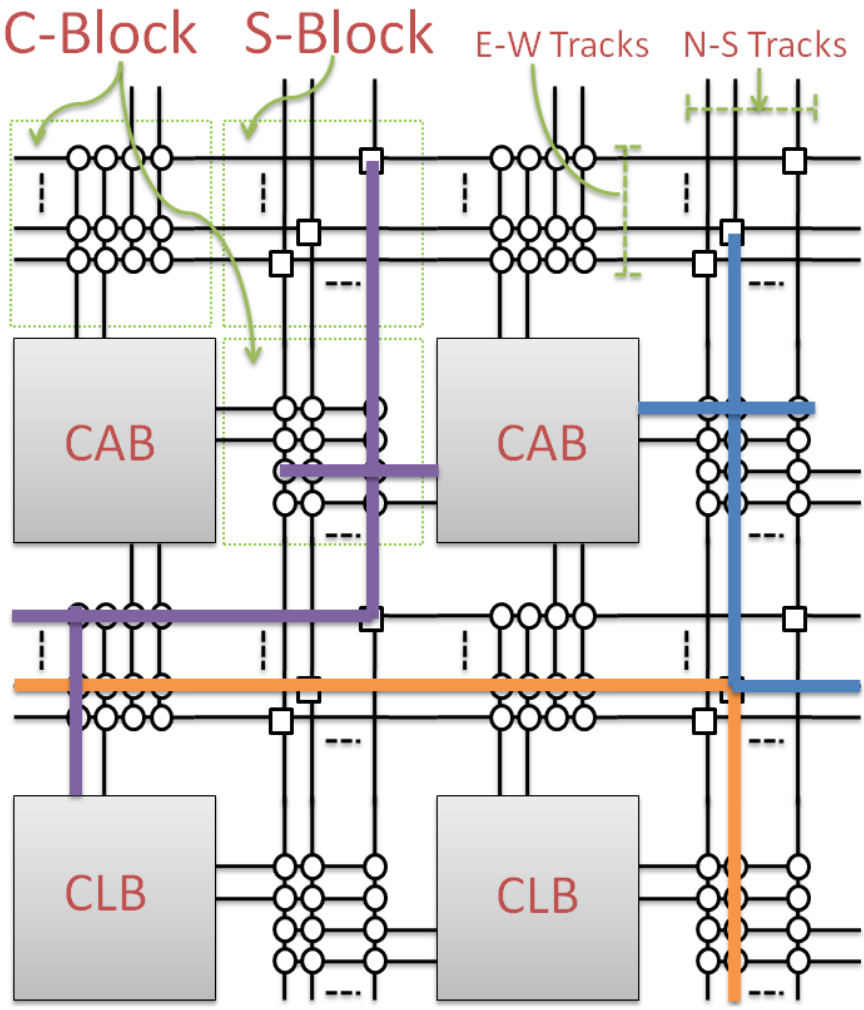
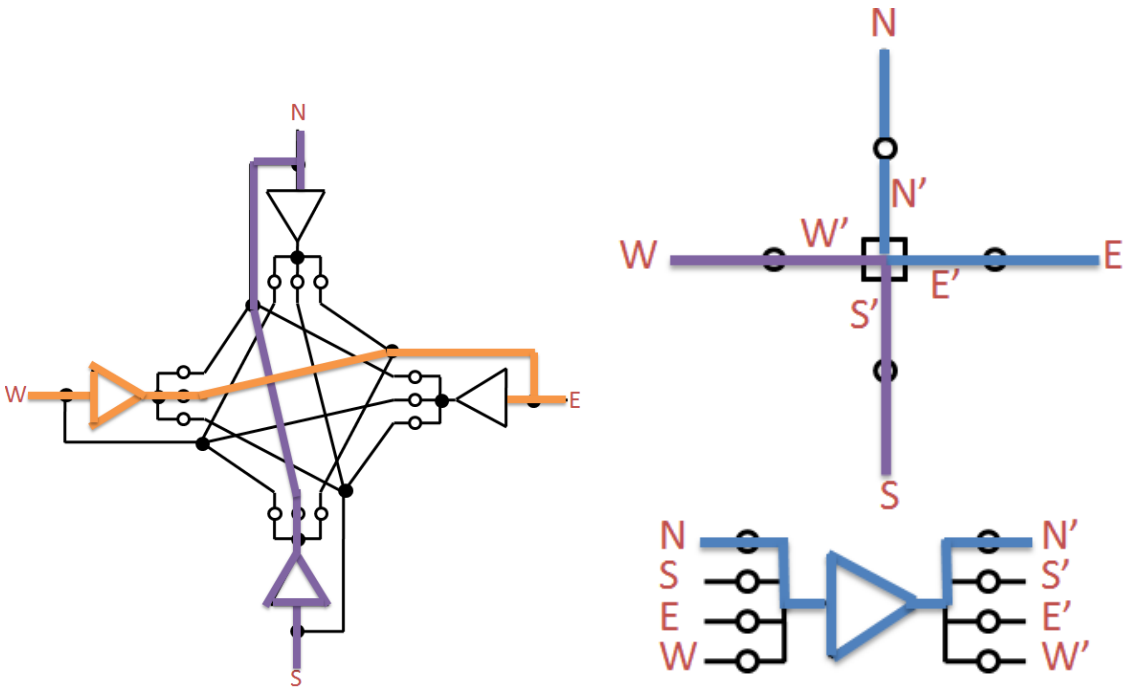


Figure 31: The global interconnect comprises vertical and horizontal track segments isolated by S-Blocks. The S-Blocks allow signals on tracks to propagate to neighbor tracks or to change directions. The C-Blocks provide connectivity from the global tracks to the primary inputs and outputs of the CLBs and CABs. Examples of allowable routings shown highlighted.



(a) (b)

The s-switch topology used in the digitally buffered s-switches. b) The analog buffered s-switch topology. Some examples of allowable routings shown highlighted.

The standard s-switch is implemented as shown in Figure 28e, which passively passes both analog and digital signals. Every s-switch is of this passive type except for the bottom left ones on the first track.

These s-switches on the bottom track are buffered. Each digital tile's S-Block has a single digital buffered s-switch and each analog's has a single analog buffered s-switch. Two different buffered s-switch topologies can be seen in Figures 32a and 32b. Both circuits are bi-directional, and allow for the same direction choices of signal propagation as the passive s-switch. The first circuit uses significantly less switches, has less internal parasitics per track, forces all entering signals to leave buffered, and requires four buffers. The second circuit is basically a passive s-switch with the ability to insert a single buffer on the input from one of the directions. In general, the first topology will be faster, but larger than the second topology. Because the analog buffers (a 9T floating-gate programmable OTA based unity-gain buffers) are much bigger than the digital ones (two-stage inverter chain) we chose the second topology

for the analog buffered s-switches and the first topology for the digital buffered s-switches.

Array size	27x8: 108 digital tiles and 108 analog tiles.
Chip IO	33 generic IO pads, 11 digitally buffered bi-directional pads
Devices per CAB	2 FG-OTAs, 2 TGATEs, 2 Capacitors, 2 FETs (nFET or pFET), 2 MITEs
Devices per CLB	4 BLEs
CAB / CLB I/O	8 inputs, 4 outputs
BLE	3-input LUT, routable clock and reset, reconfigurable for asynchronous adders
C-BLOCK	11 Total tracks, all of segment length 1, fully connected connection blocks
S-BLOCK	diagonal with 1 digital or analog buffered s-switch per tile on the first track
Process	CMOS 0.35um Double-Poly, 4-M
Voltage	2.4V at runtime

Table 2: FPAADD specifications

Table 2 contains a list of specific parameter values used in the FPAADD.

5.2.2 Interconnect Comparison

The CAB devices, floating-gate design, and floating-gate programming infrastructure were all derived from the RASP 2.9a chip, a next generation FPAA from the line developed by Hasler *et. al.* [28, 27, 5]. Significant differences from the RASP 2.9a include the choice of a Manhattan style global routing architecture, and a feedback output local interconnect scheme. The global interconnect has significantly less parasitics over short distances than the RASP's global scheme, where global tracks span the entire length of the chip. There are buffers in the global interconnect whereas the generic RASP line contains none. The local interconnect of the FPAADD has 68% lower parasitic capacitance and 50% less parasitic resistance between routed CAB devices in the local interconnect (devices in the FPAADD can be connected with one switch, but in the RASP chips require two at minimum) at the cost of slightly

decreased routability described earlier. This leads to a 4x improvement in bandwidth of signals routed in the local interconnect over previous RASP based FPAAs. In the RASP line, CAB devices were disconnected from the routing infrastructure during program time with large transmission gates in order to not expose the devices to injection level programming voltages, but with careful circuit consideration, these can be removed in almost all cases.

The global interconnect scheme as well as the local interconnect and CLB devices draw heavily from standard Manhattan style FPGAs [29, 30]. Ignoring semi-arbitrary design decisions regarding architectural parameters, such as number of tracks, cluster size, placement of buffers, etc. the digital tiles look very similar to previously made FPGAs. The biggest difference being replacing the switch elements and SRAM with programmable floating-gate pFET transistors[28, 27, 5].

Floating-gates are very similar in operation to non-volatile technologies such as EPROM, EEPROM, and FLASH; various FPGAs and CPLDs have been built using these technologies [31],[32],[33]. The floating-gates transistors in the FPAADD are built in a standard CMOS process. They have a higher dynamic range of programmed voltage leading to significant performance increases in power, speed, and signal integrity at the cost of density compared to conventional EEPROM and FLASH devices.

In [31], they claim that switching from using the EPROMs as the actual switch to simply using the EPROMs to control the gate of CMOS devices, that a 10x speedup was achieved. This is similar to the problem that pass-transistor logic, often used in FPGAs, face when trying to pass a logic-level (VDD for nFETs and GND for pFETs) that causes the devices to enter subthreshold before completely passing the signal. This results in logic level high voltages of about one threshold voltage less than VDD after reasonable amounts of time, usually leading to speed degradation and an exponential increase in leakage current in gates driven by these logic levels.

Because the floating gate voltage can be programmed to higher than one threshold voltage above VDD, the switches stay in above threshold while passing the whole rail-to-rail voltage. Small signal resistance sweeps in [27] show floating-gate switches being as good as transmission gates but at half or better parasitic capacitance.

5.3 CAD Software for the FPAADD

Much work has been done in the realm of synthesis for FPGAs; many software packages are available from industry and the open-source community alike. The field of synthesis for FPAAs, however, is far from mature. While the algorithms for placement and routing certainly have application to FPAAs, what does not translate so well are the cost functions (other than trivial ones like area and routability) to evaluate the desire-ability of routable solutions. FPGA synthesis is largely timing driven, where propagation delay models are used to identify the worst case delay of the critical path (further effort can be spent to then reduce the amount of devices on non critical paths for power optimization). While line delays certainly have some application to analog circuits, they are by no means the appropriate metric for all circuit nets.

In [34] the authors successfully apply standard placement and routing algorithms to map analog circuits to FPAAs with global parasitic reduction being the metric of choice. Next, extraction of parasitic elements is performed and back annotated to the initial input spice netlist for simulation, with fitness evaluation and iteration up to the user. The strategy in [35]is to partition the mixed signal reconfigurable system into the digital and analog subcircuits at data converter interfaces and apply different cost functions to each. Models for SNR estimation are developed that start with known device SNR and its degradation by connection topology of interconnect: cascode, fan-out, fan-in, and feedback. Bandwidth is also estimated using the data converter's Nyquist criterion as the bottleneck.

None of these approaches take into account the appropriateness of applying different cost functions to different net types. For instance, an algorithm that places negative weight on average parasitic capacitance of all nets will inefficiently try to reduce parasitic capacitance on nets that are insensitive to it, like internal nets of DC bias generators.

The software suite, Verilog To Routing (VTR), was extended and modified to perform placement and routing on the FPAADD. As of this writing, the flow is completely area driven.

5.3.1 Verilog To Routing

VTR is an open source, academic software suite that given an input Verilog circuit description and an input FPGA architectural description, performs synthesis and place and route (Figure 32). The suite consists of the following programs: ODIN II, which performs logic synthesis to standard cells (in this case, LUTs, FFs, and macro functions) [36], ABC which performs logic optimization [37], and T-Vpack and VPR which perform packing of LUTs and FFs into CLBs and then placement and routing [38].

While the flow supports synthesis of Verilog to the standard FPGA building blocks, it also supports the targeting of larger functions that may exist as dedicated hardware blocks on a heterogeneous FPGA. For instance, it is common to include hardware adders or multipliers in FPGAs as the synthesis of these rather common functions are often the bottlenecks in an FPGA implemented circuit design.

The support of black boxes made VTR a very attractive starting point in creating a software chain to provide placement and routing on the FPAADD. Digital circuitry could be synthesized all the way from Verilog while the analog circuitry could be treated as black boxes and simply placed and routed.

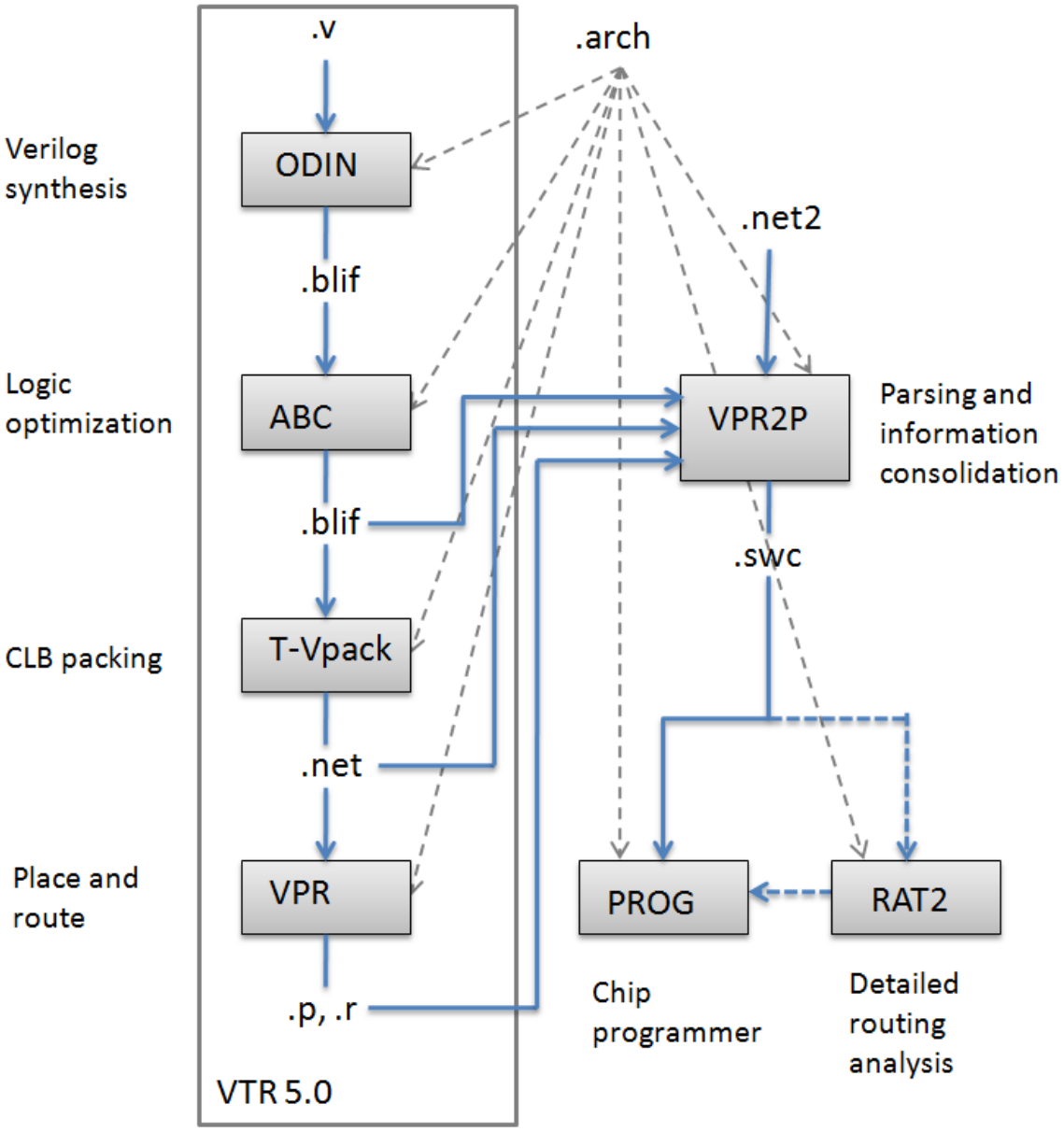


Figure 32: The software stack used for programming the FPAADD. From the VTR flow: ODIN takes an input Verilog file and performs logic synthesis targeting LUTs, FFs, and macro function blocks. ABC performs logic optimization. T-Vpack clusters LUTs and FFs into CLBs. And VPR places and routes the result. VPR2P takes an input describing the internal configuration of the CABs that are treated as black boxes in the VTR flow, and all of the intermediate outputs of the VTR flow, and creates a switch list. The switch list can be directly programmed or analyzed and modified by the detailed routing analysis tool, RAT2. All programs in the flow take various pieces of architectural descriptions of the target system.

5.3.2 Routing on the FPAADD

While VTR will route circuits to arbitrary architecture graphs, it also supports a robust and scalable XML based architecture description language for quick graph building. Since the FPAADD was designed with a Manhattan style global and local interconnect scheme, describing the FPAADD in the VTR architecture language was relatively straight forward. Only a few minor modifications to VTR 5.0 were necessary.

The current flow starts with the circuit input as a blif and a net2 file. The blif file contains all of the digital circuitry as described as netlists of LUTs and latches with black boxes for the analog circuits. T-Vpack then packs the digital circuits into CLBs and the CABs are already prepacked in the net2 file. VPR then places and routes the CLBs and CABs.

The program VPR2P was written to take all of the intermediate file outputs of the VTR flow, consolidate the information, fill in the blackboxes with information from the net2 file, and then to translate the information into the corresponding physical switch locations on the FPAADD. The output is a row column switch list that is the input into our programming software. It also handles chip and board communication as well as the algorithms for erasing and programming the floating-gate memory elements.

Since VPR is not concerned with local interconnect, as routing at that level is deterministic, the GUI does not show the internals of the blocks. In order to analyze the detailed routing solutions (global routing and local routing) and to provide for a way to set and unset switches by hand, the RAT2 tool was created.

The RAT2 is a simple program written in MATLAB that can read in a switch list, display the routing solution, modify by means of a point and click interface the switch list, and dump out a switch list. This switch list can then be used to program the chip.

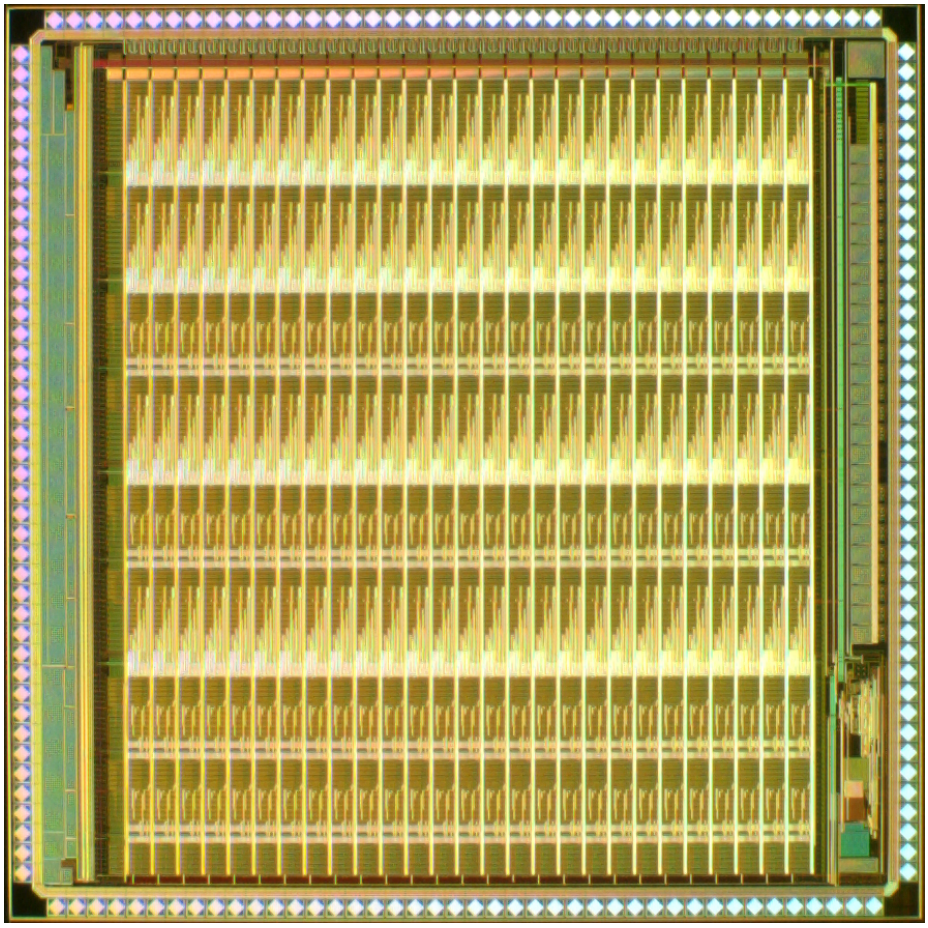


Figure 33: Die photo of the fabricated FPAADD.

5.4 System Verification

The FPAADD as described in Section 5.1 was fabricated in a standard double-poly, single n-well, 4 metal CMOS 0.35um process. A die photo of the FPAADD is shown in Figure 33. All reported data are taken from the fabricated chip. The system is operated at 2.4V during run time, as opposed to 3.3V, to increase retention of the stored charge on all floating-gate transistors [5].

All CAB and CLB devices, as presented in Table 2, are verified to be functional, via successful interconnect routing to I/O pads; global interconnect, local interconnect, and interconnect buffers are all working as expected. Simple circuits have been built: XOR gates and full-adders implemented in the CLB floating-gate based LUTs,

asynchronous adders generated from FFs and LUTs, MOSFET threshold and characterization data extracted from transistor devices in the CABs, and ring oscillators built out of the buffered global interconnect. Verification and programming of the the floating-gate transistors were performed and found to be similar from results reported in [5]. The design and layout of components for the CAB were re-used from previously designed FPAAs and performance metrics were found to be comparable to published literature in [5].

To evaluate the performance of the interconnect network, we measured delay as a function of routing distance (i.e. interconnect stages). Ring oscillators were implemented to perform this measurement, each interconnect stage being a C-Block and S-Block. Both buffered and unbuffered digital tracks are measured. In Figure 34, oscillator period is plotted as function of the number of stages. As expected, the delay of the oscillators using non-buffered tracks increases quadratically with the number of stages as is typical of RC ladders. The delay of the buffered tracks increase linearly. The delay of moving from one tile to the next through a digitally buffered s-switch is 1.6ns. Using a similar method, the BLE to BLE delay was measured to be less than 7ns.

5.5 System Examples and Measurements

Previous FPAAs have been used to build continuous time filters, vector matrix multipliers, AM receivers, analog speed processors, among others [4, 5]. The reconfigurable and mixed-signal nature of the FPAADD allows the user to address a variety of applications from pure analog to mixed-mode to pure digital including FPAA applications in previous literature. Two example system applications have been built to demonstrate the configurability and performance of the FPAADD: a VCO-based ADC and a 2^{nd} order low-pass sigma delta modulator.

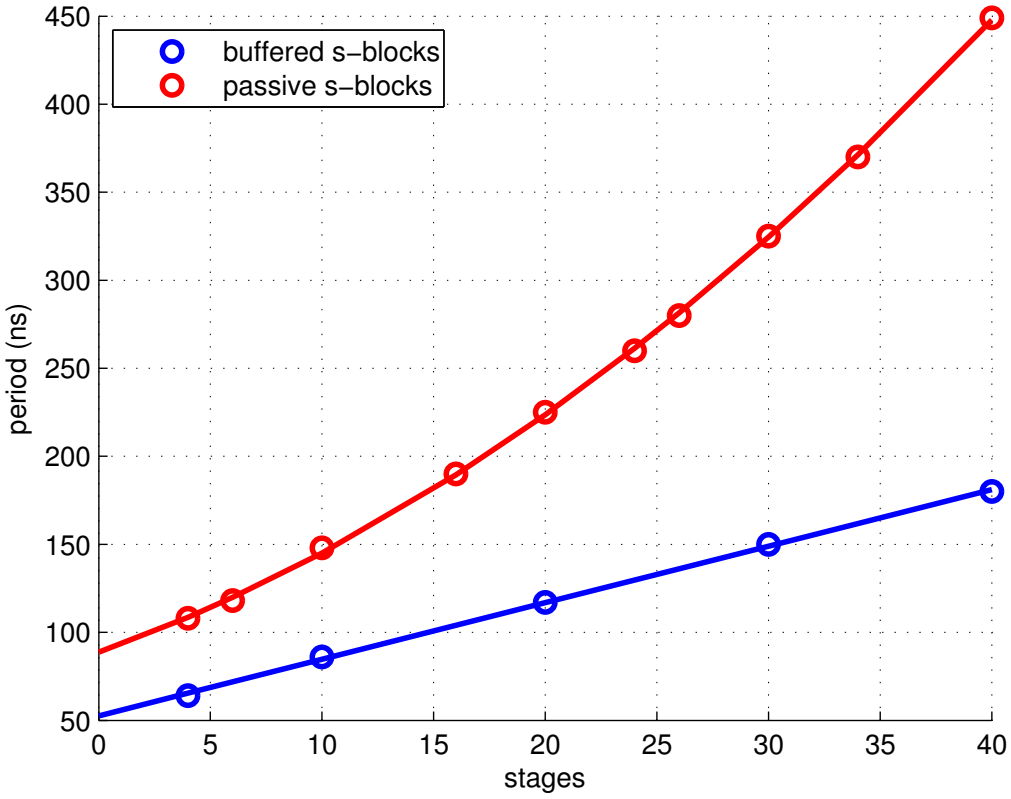


Figure 34: Ring oscillator period versus number of additional interconnect stages (s-block to s-block) for digitally buffered and passive s-blocks. The incremental delay due to a digitally buffered s-block is 1.6ns.

5.5.1 VCO ADC

An 8-bit VCO-based ADC was built in the FPAADD as shown in Figure 35. The voltage controlled oscillator was built using discrete transistor CAB components; the asynchronous counters, state machines, and registers were built out of the CLBs. Figure 36 shows the frequency versus control voltage plot of the VCO. The linear dynamic range of the VCO was measured to be from 0.18 MHz to 7 MHz. The digital back-end was clocked externally at 2 MHz. However, the back-end was operational up to 18 MHz.

The ADC was measured to have no missing codes, and its operation can be seen in Figure 37 for a 200.137Hz, $0.4V_{PP}$ input sine wave applied at V_{in}. INL and DNL data is not presented due to the non-linearity inherent in VCO based ADCs. The non-linearity of the ADC is due to the following effects: the voltage (V_{in}) to current converter is a simple nFET operated in sub-threshold, so an exponential voltage to current conversion is expected, while the rest of the VCO (a current controlled oscillator) performs a linear conversion of input current to frequency. The digital back-end counts the number of CLK_{ADC} transitions per VCO output pulse (V_O), giving a measure of the period for V_O. Using expected circuit behavior, the input is reconstructed from the output by fitting it to the following equation:

$$-ln[aT_{out} + b] = V_{out} \tag{14}$$

where T_{out} is the measured output, a and b are terms lumping sub-threshold parameters of the input V-to-I input stage and the linear current controlled oscillator stage. The signal is then reconstructed from the ADC output and shows the circuit to be in excellent agreement with expected circuit behavior, as seen in Figure 37.

The VCO based ADC system consumed a total of 10 tiles (four analog and six digital) representing 4.6% of the total number of tiles in the FPAADD array. The

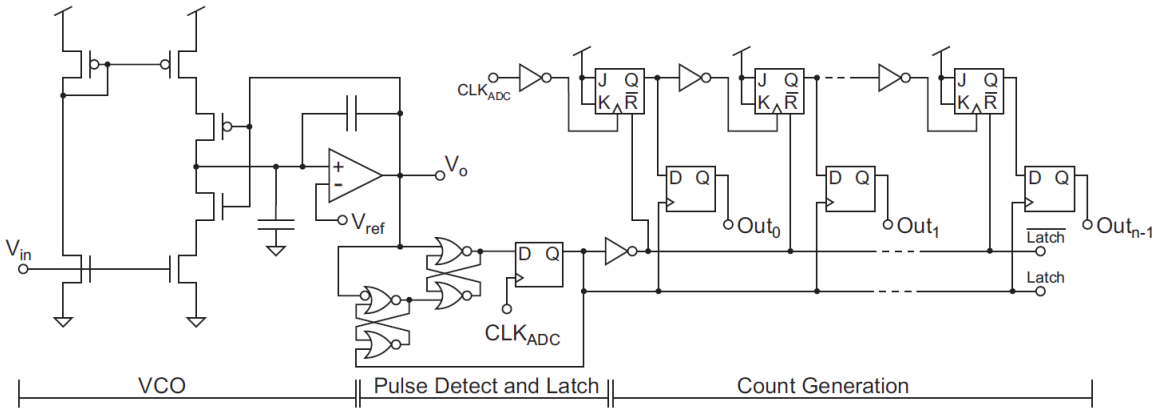


Figure 35: An 8-bit ADC built on the FPAADD. a) Block Diagram: A current or Voltage Controlled Oscillator's (VCO) output period is measured by a digital backend. b) Timing diagram for the circuit's operation. c) VCO, pulse detection circuit and state machine, asynchronous counter and latches.

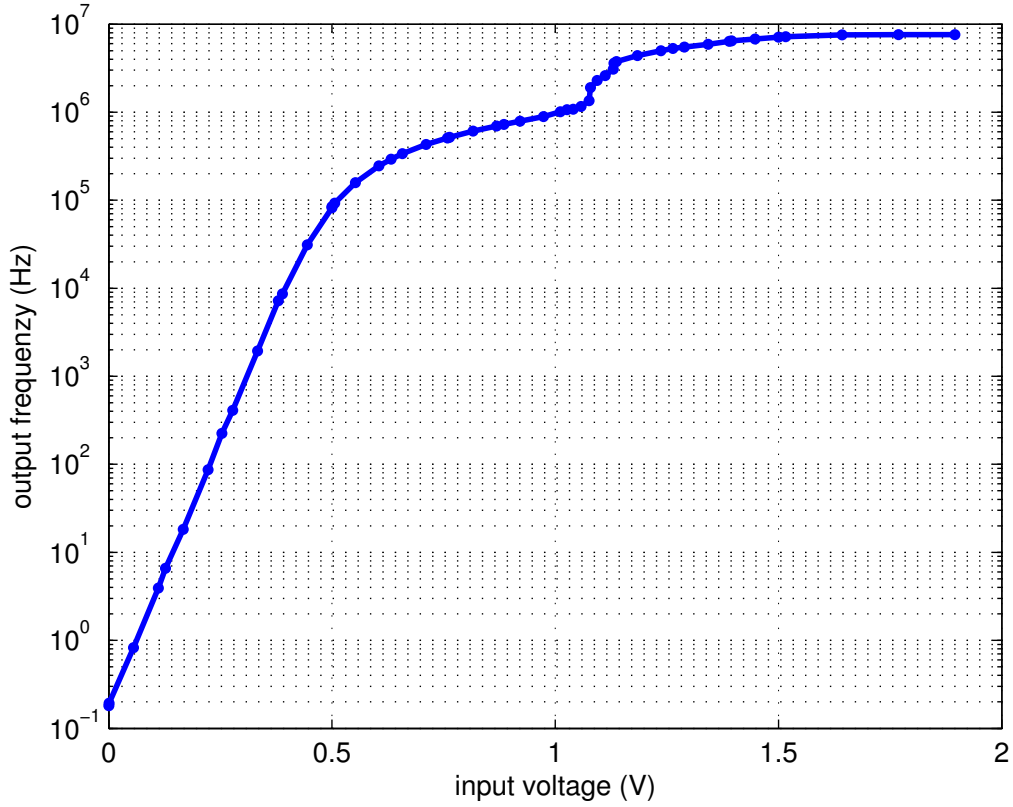


Figure 36: Measured response of the VCO over varying input voltage.

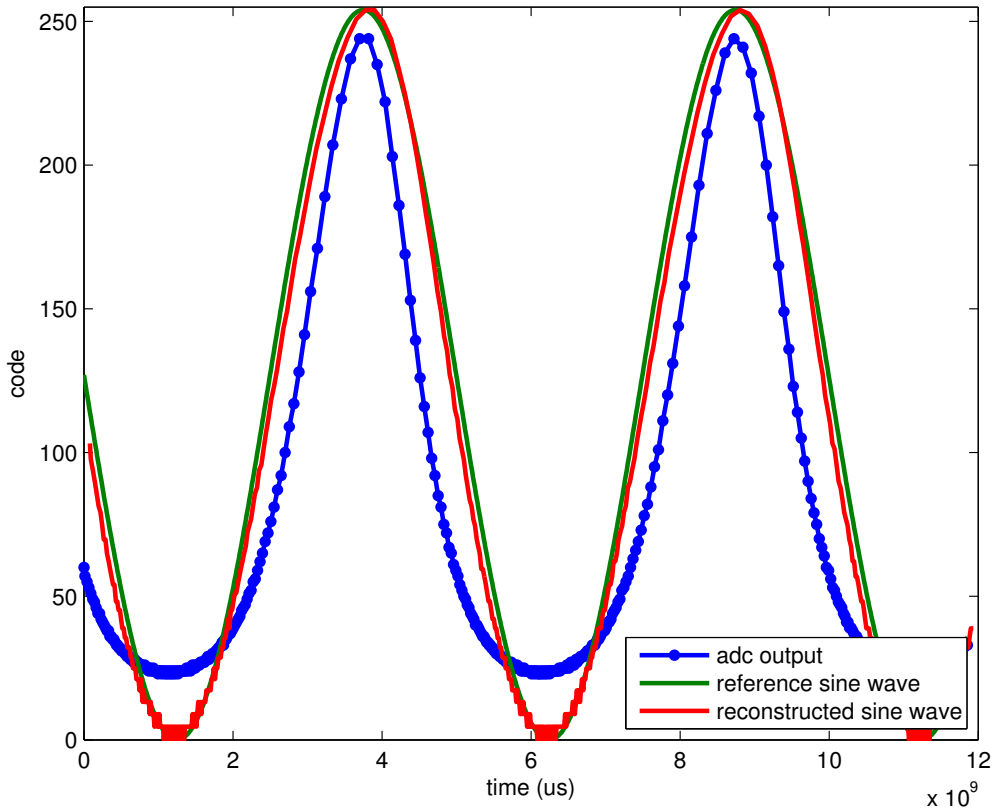


Figure 37: 8-bit VCO based ADC digital output (dotted line) for a 200.137Hz input sine wave of $0.4V_{PP}$ and the reconstructed input signal.

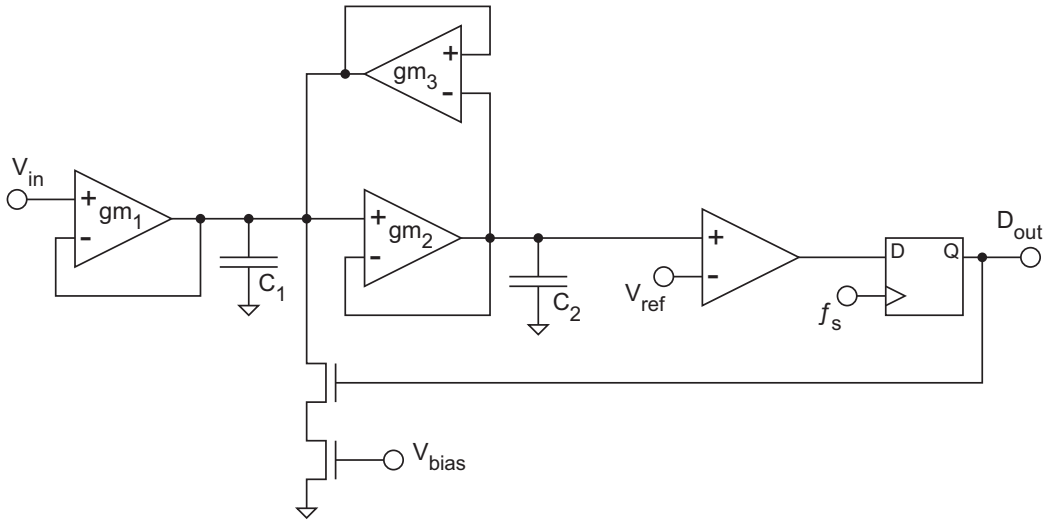


Figure 38: A 2^{nd} order sigma-delta modulator with 1-bit DAC feedback.

percentage of device utilization within the six digital tiles was 88% while the utilization within the four analog tiles was 23%. Low element utilization of the CAB is due to the heterogeneous nature of the devices present within the CAB. The VCO used primarily discrete transistors found in the CAB along with an OTA and 2 capacitors leading to the low utilization value.

5.5.2 Delta-Sigma Modulator ADC

Figure 38 depicts the system diagram of a 2^{nd} order low-pass sigma delta created in the FPAADD. The low-pass filter was built using components from 2 CABs, and a single CLB is utilized for the D Flip-Flop. The poles of the loop filter are designed to be located at zero. The sigma-delta modulator has a measured SNR of 24.1 dB and SFDR of 39.2 dB at a bandwidth of 20 kHz and over-sampling frequency of 2.5 MHz. Figure 39 is a 32k FFT of recorded data taken from the FPAADD at the previously stated input and sampling frequencies. Insufficient gain of the loop filter is the probable reason for lower than expected SNR. Further optimization of the loop filter is required to increase the SNR.

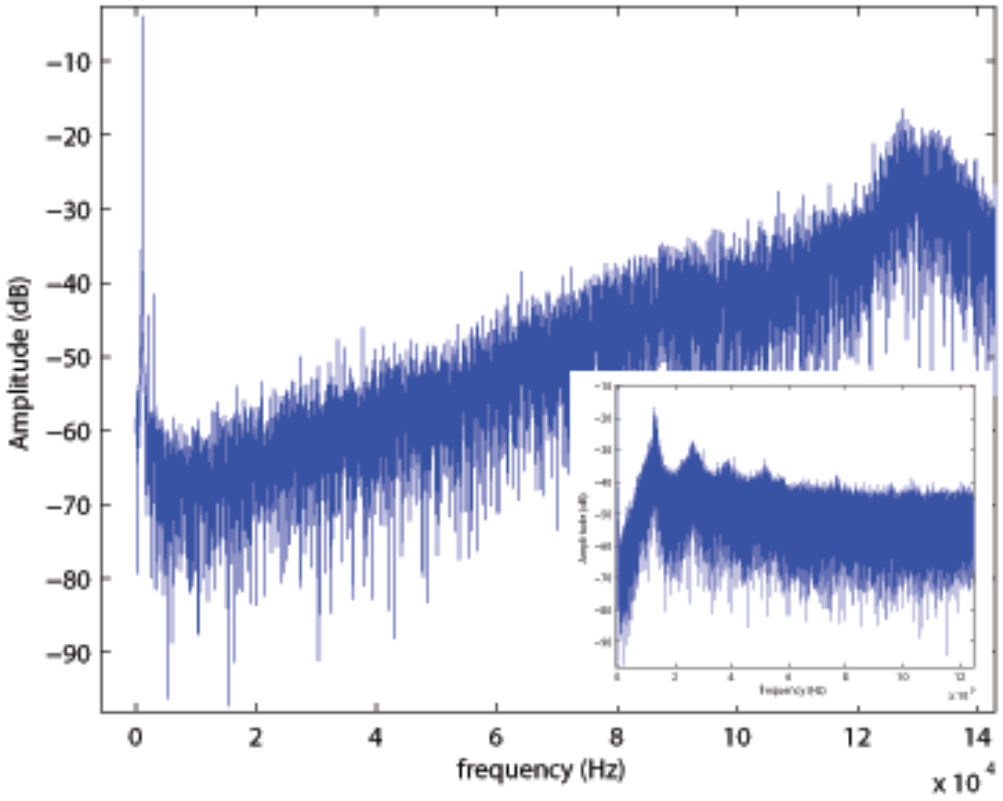


Figure 39: A 2^{nd} order sigma-delta modulator with 1-bit DAC feedback. Measured power spectrum for an input of 1.0478 kHz at 2.5 MHz oversample frequency.

5.6 Conclusion

A mixed-signal heterogeneous tile array (FPAADD) of CAB and CLB components has been built and presented. Verification testing of the system was performed at the component, tile, and system level. Initial results of the FPAADD display 7ns BLE to BLE performance and 1.6ns buffered tile to tile delay. Oversampling ADCs were implemented to test the functionality of the tile array and show the reconfigurable nature of the chip. The next stage of research will further characterize the FPAADD with emphasis in system scalability, power and noise analysis, and optimum partitioning of analog/digital functionality. This will allow realization of larger systems that take full advantage of all the computation properties. The goal of the FPAADD is a bridge towards embedded systems containing the reconfigurability of a FPAA and digital processors, resulting in an embedded single chip reconfigurable solution to implementing complex systems.

CHAPTER VI

SYSTEM-ON-CHIP FPAA

The FPAADD system introduced the concepts of a fine grained, mixed-mode, heterogenous tiled array of reconfigurable systems designed to support a modern synthesis and place/route toolchain. However, the chip still required significant external off-chip resources, namely, a microprocessor to perform the floating-gate programming algorithms and communication with the user PC or other external systems. To enable further integration and add more functionality to our reconfigurable systems, the FPAADD design and the RASP 2.9v was used as the basis for a new generation system called the RASP 3.0, as it will be the third generation of RASP chips [28, 5, 39].

6.1 System Architecture

Figure 40 shows the basic system architecture of the RASP 3.0 chip. Taking the FPAADD concepts of analog and digital tiles, a manhattan style light weight interconnect scheme and the volatile shift registers from the RASP 2.9v, the 3.0 chip incorporates an embedded microprocessor, memory, digital peripherals, low power array of DACs, a 32k analog memory bank, and various fixes/tweaks to the floating-gate programming infrastructure. The new RASP 3.0 chip is considered to be the first RASP System-On-Chip (SoC) implementation.

The processor is a synthesized MSP430 core variant from the OpenMSP430 project. The core is instruction set compatible with the MSP430 line of micro controllers from Texas Instruments. Compilation of code is performed using an open source gcc variant for the MSP430. Including the openMSP430, the back-end includes, 2x16k SRAM, general purpose input/output peripherals, serial-peripheral interface

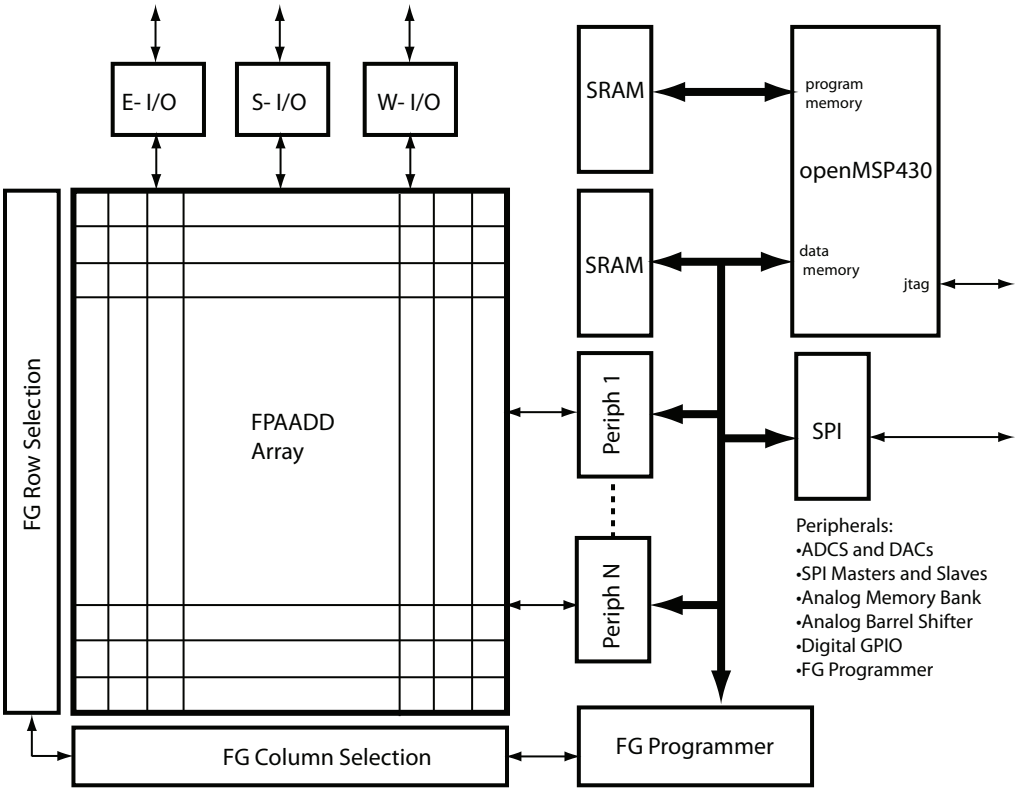


Figure 40: An FPAADD with integrated processor for on-chip floating-gate programming control and runtime computation and datapath control

(SPI) bus, timing block for a volatile analog memory block, floating-gate programming digital logic/state machine, and miscellaneous control logic. Communication between the CPU, peripherals, and access to the FPAADD array is accomplished via a memory-mapped I/O bus. Communication is provided by an on-chip SPI bus and the microprocessors built-in debug 2-wire interface. The FPAADD array is modified to support direct and indirect floating-gate switches, volatile switches, and shift registers.

The shift registers are treated as generic devices in the CABs, with all shift register control signals being locally routed inputs, Fig. 41 depicts volatile switch realization in the RASP 3.0 system [40]. This allows one to vary the number and depth of any shift register through programming, as well as immense flexibility in the detailed control of shift register operation: one registers output could clock another shift register, or clocking could be created from synthesized digital state machines, or be driven directly by SPI peripheral blocks controlled by the processor. The flexibility of the shift registers and usage as a generic CAB device, allows it to be utilized with high efficient into high level synthesis tools. Target applications for the RASP 3.0 can include image transforms, synthesizable data converters, PLLs, frequency synthesizers, PWMs, and analog data paths with digital control.

6.2 RASP 3.0 Synthesis, Place and Route Tool Flow

The RASP 3.0 tool flow is based upon the flow created for the FPAADD, in particular the usage of the VTR/VPR software [25, 41]. The front-end for the new flow starts at Xcos, which is an open source graphical dynamical model simulator similar to Simulink from Mathworks. The Xcos based graphical front-end runs on Scilab, again an open source software similar to Matlab from Mathworks. Figure 42 shows the gui used in the RASP 3.0 tool flow to design circuits and systems for the chip. In particular, Fig. 42 depicts a simple ramp generator which may used as part of a ramp

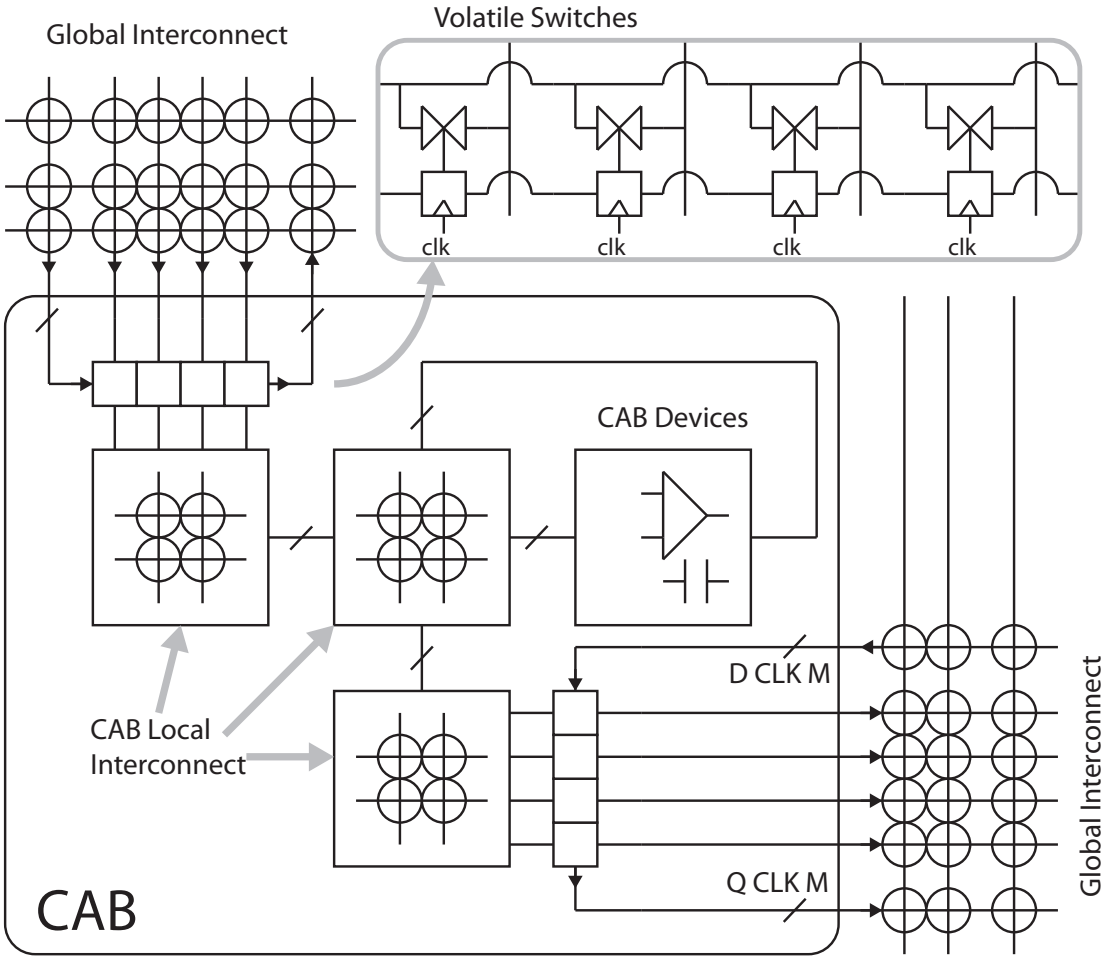


Figure 41: Simplified schematic of the volatile switches in the RASP 3.0 chip. The control signals and data lines for the volatile switches are themselves routed signals from the tile array.

ADC.

After the user has designed a circuit/system in Xcos, the custom tool software creates a blif file [42]. The blif file, as shown in Fig. 43 is used as the input for VPR to determine basic element (i.e. opamp, capacitor, logic gates, flip flops, etc.) placement and routing of the circuit nets. At the blif level, we have implemented support for flip flops with clock multiplexers. In traditional FPGA architectures, the clock for the flip-flops in the BLEs are the same global clock. Starting with the FPAADD and enhanced in the RASP 3.0, the clocks for the flip-flops are chosen between a global clock, routed clocks from the tile array, or the output of a previous BLE (to enable efficient counters). The ability to route clocks from the tile array into the CLBs enables the RASP 3.0 to create digital system with different clock domains, asynchronous logic circuits, and efficient counters. This is a key capability and difference from previously cited examples of FPAAs and hybrid reconfigurable systems.

The blif is the input into VPR to generate packing of the tile CAB/CLB elements, signal routing between tiles, and to/from the chip I/O. Figure 44 shows the graphical interface showing the final routing solution generated by VPR. After packing and routing by VPR, the resulting files are parsed by the RASP 3.0 custom software tools to output the final list of floating-gate addresses. This list of addresses is similar to the programming list for FPGAs. It dictates which floating-gate devices to enable for routing of signals and accurate programming for biases, VMMs, or arbitrary analog weight storage among other possibilities. Figure 45 depicts an examples from a switch list for the ramp generator shown in Figure 42. The last step in the tool flow is programming of the RASP 3.0 using the generated switch list and verification of the programmed addresses.

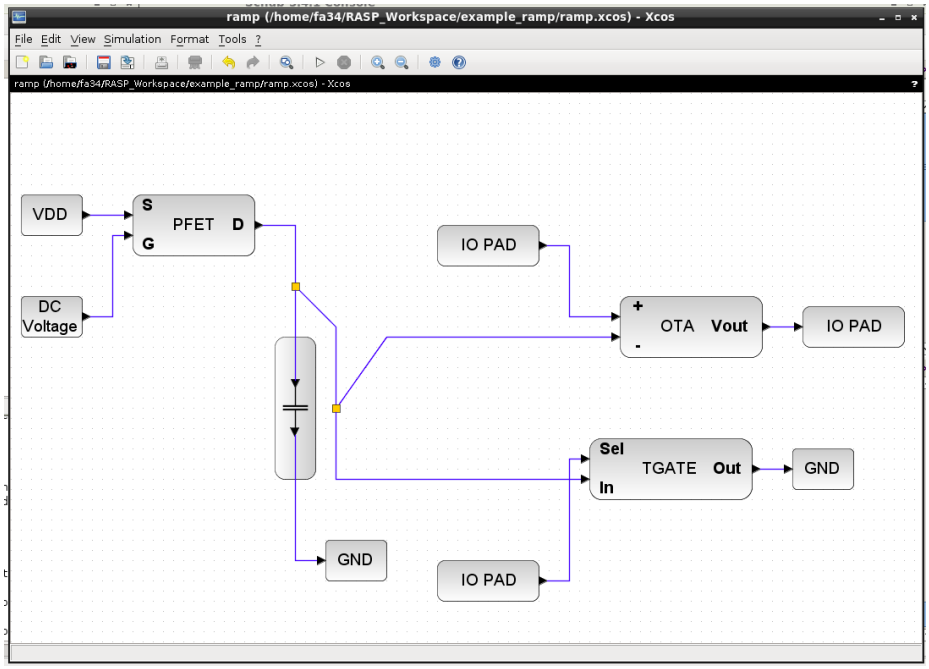


Figure 42: Xcos interface for the RASP 3.0. Circuits and systems can be created from basic CAB/CLB blocks and larger macro blocks. Shown here is a very simple ramp generator which can be used in a ramp ADC.

```
.model newclb
.inputs net1 net2 net3 net4 rst
.outputs out1
.clocks

.names net1 net2 net3 net4 lutout
00-- 1

.subckt latch_custom D[0]=lutout clk[0]=net4 reset[0]=rst Q[0]=dffout

.names dffout net1 net2 net3 out1
1--- 1

.end
```

Figure 43: An example of the intermediate blif file created from the Xcos model and used as the input for VPR.

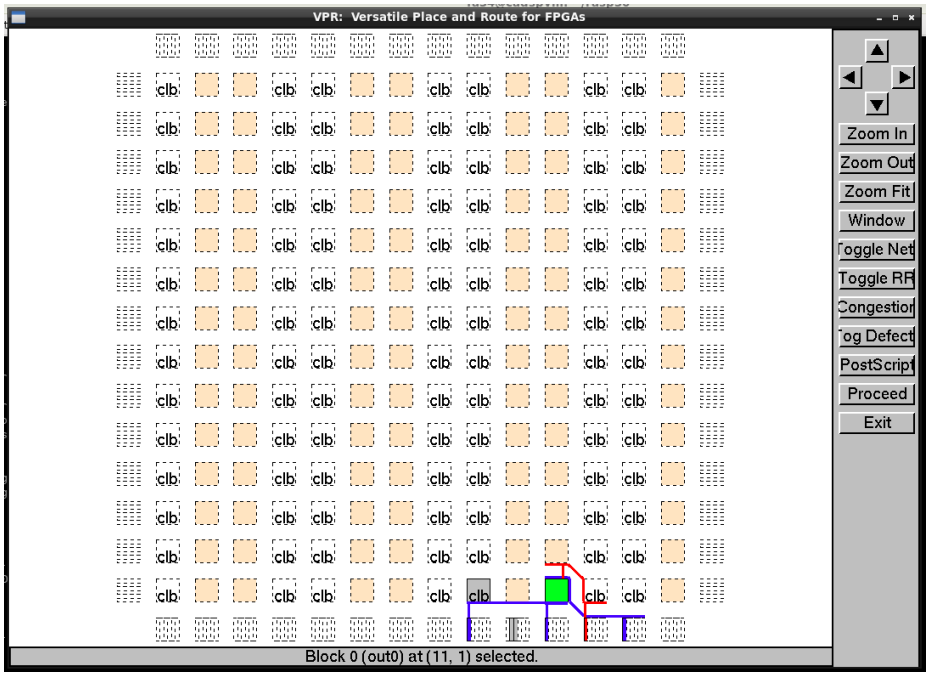


Figure 44: The ramp generator circuit from Fig. 42 as packed and routed using the RASP 3.0 tool flow and being shown using VPR. The VPR tool performs packing of the basic blocks/macroblocks into the CAB/CLB. It also performs routing of circuit nets between tiles and the chip I/O.

```
401 272 0 0
413 400 0 0
425 673 1 0
379 432 0 0
391 529 0 0
123 272 0 0
136 448 0 0
174 448 0 0
208 448 0 0
238 448 0 0
272 448 0 0
310 448 0 0
344 448 0 0
374 448 0 0
```

Figure 45: The final output of the tool flow is a text file containing a list of floating-gate addresses.

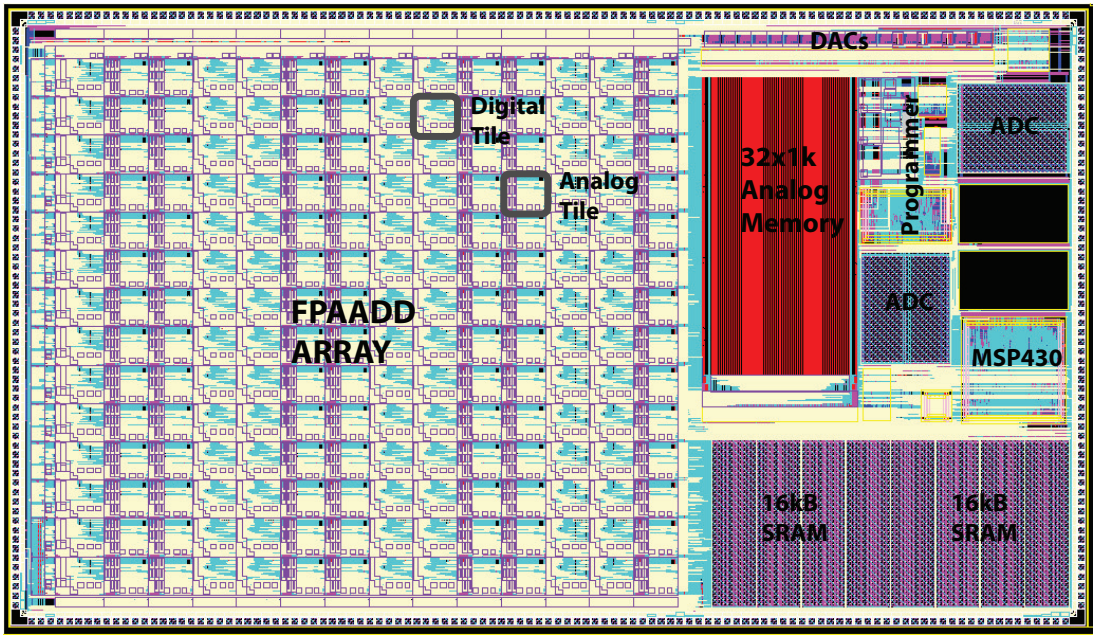


Figure 46: Layout of the RASP 3.0

6.3 Measured Results from the RASP 3.0

The RASP 3.0 was fabricated in a 0.35μm CMOS process and the chip size is 7mm x 12mm. Figure 46 depicts the layout of the RASP 3.0 along with highlights of the various elements compromising the system.

A simple 1^{st} order $G_m - C$ filter was constructed to show basic operation of the RASP 3.0 chip along with verification of the new software/tool flow. Using the same components, the filter response was tuned via modification of the bias current generated from a floating-gate transistor, similar to methods shown in Chapter 4.

Figure fig:ramp is the output response of the ramp generator from Fig. 42. The input to the generator is an enable signal, while the output of the ramp was measured. The non-linearities of the ramp are due to the use of bias current source with a low output impedance. The basic ramp generator shown can be modified to generate a more precise output using the available components in the RASP 3.0.

The digital CLB was verified by creating an arbitrary logic function from Verilog via the RASP 3.0 tool flow. An arbitrary logic block performing the following action

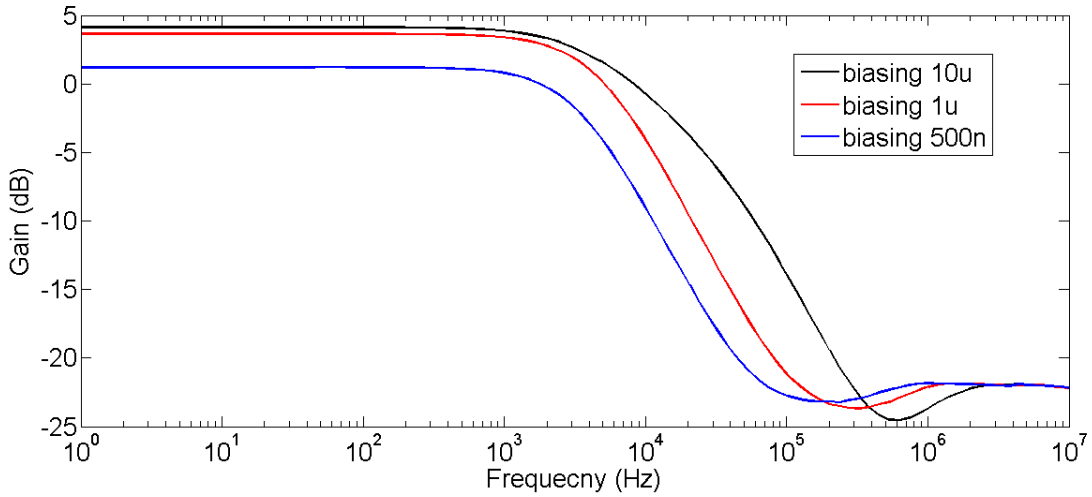


Figure 47: Frequency response of 1^{st} order $G_m - C$ filter with various bias currents programmed via floating-gate transistors.

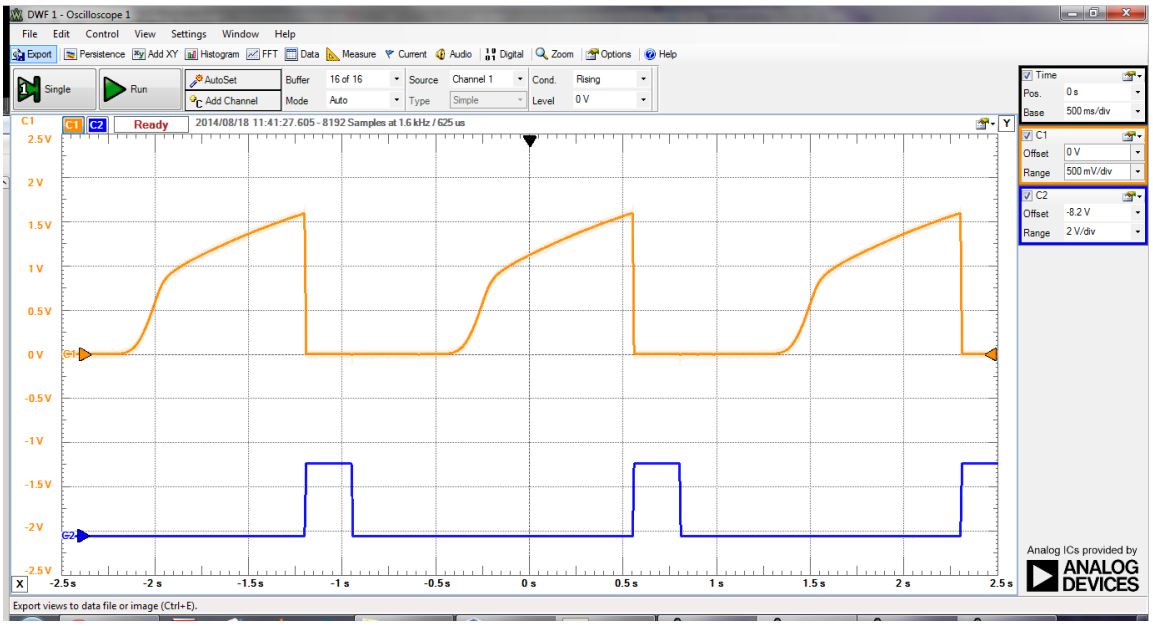


Figure 48: Output response of the ramp generator from the Xcos model of Fig. 42.

was designed:

$$
\begin{aligned}
X &= \overline{ab}c + ab\bar{c}, \\
Y &= Z@Clk, \\
Z &= \overline{Y}.
\end{aligned}
\tag{15}
$$

The logic takes a 3 bit input, performs an arbitrary addition, using one BLE. The output X is sent into a D Flip-Flop clocked by signal Clk. The output of the flip-flop, Y is inverted to obtain Z, using a second BLE. Figure 49 is the resulting output of Eqn. 15 compiled and programmed on the RASP 3.0 using the tool flow. Signals 11, 10, and 9 are a, b, and c, respectively. Signal 8 is Clk and the output Z is signal 7 in Fig. 49. The resulting output signal is what should be expected given the inputs and Eq. 15. Given the above examples, we have experimentally measured and verified the RASP 3.0 working using our tool flow. Although, the examples shown are basic in nature, they are building blocks for larger and complex systems.

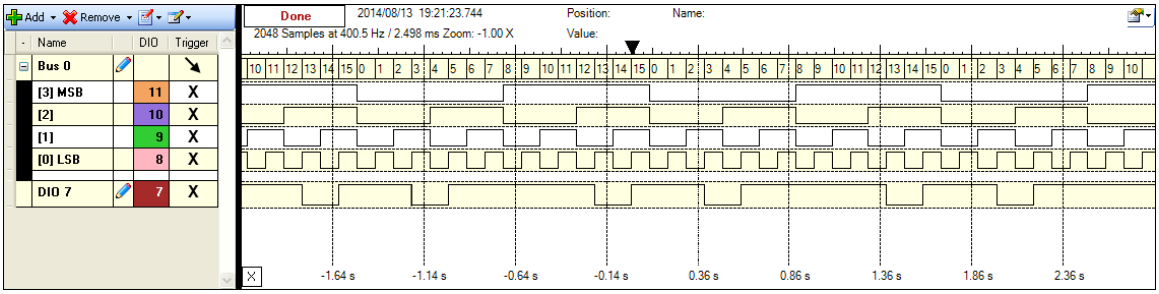


Figure 49: Response of a digital circuit created from multiple LUTs and one flip-flop using the RASP 3.0 tool flow.

6.4 RF optimized RASP 3.0

CMOS process scaling has enabled IC systems to increase performance enabling systems to operate in the RF domain. In order to obtain faster and high performance systems using reconfigurable chips (i.e. create better data converters from mixed-signal reconfigurable systems), a modified RASP 3.0 system has been designed in a

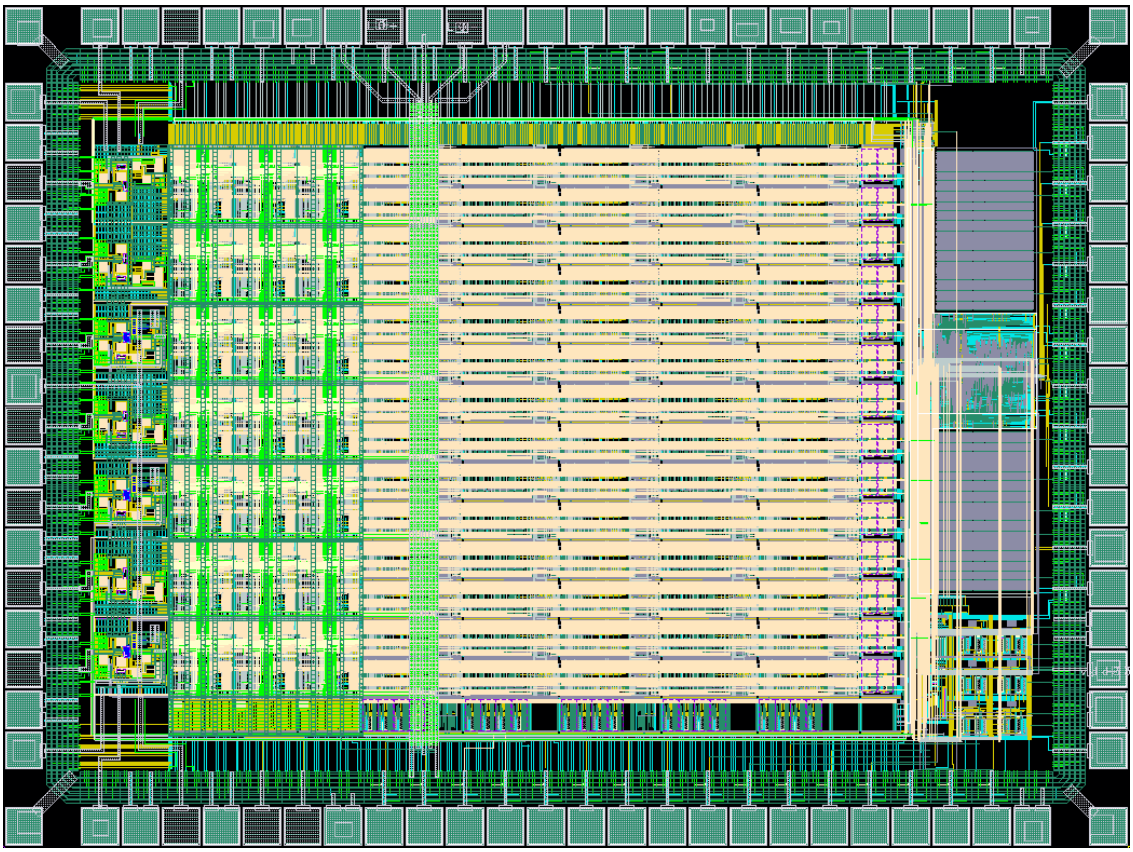


Figure 50: A RF optimized FPAA (RASP 3.0 RF) based on the RASP 3.0

40nm CMOS process. Along with the gains of density and lower power, a RF optimized tile array was added to this chip, which we call the RASP 3.0 RF. (or RF FPAA) Floating-gate devices can be used to higher system frequencies in RF/mixed-signal systems due to shrinking process nodes, as well. In Chapter 3, we have shown floating-gates working at a 40nm CMOS process node. Due to the 40nm process overall lower capacitance (source/drain, interconnect, etc.), we can assume floating-gate devices will have higher performance metrics.

6.5 RF RASP 3.0 Architecture, Implementation and Testing

Figure 50 shows the layout of the RASP 3.0 RF built in a 40nm CMOS process. Similar to the RASP 3.0, this new chip has an embedded CPU, 16k SRAM, digital peripherals for floating-gate programming, an SPI interface, and digital peripherals to enable memory mapped I/O to/from the array and external I/O pads. For high frequency (1-4 GHz range) operation and signal processing, we created an RF optimized reconfigurable array with associated RF CAB (or CRB) and optimized floating-gate switches for operation at the stated frequency ranges. The RF CAB contains high bandwidth OTAs, an active mixer, a passive mixer, and a capacitor bank. This "RF front-end" interfaced with the regular (baseband) array using a simple one-to-one global interconnect mapping. Along with the optimized front-end, we added Low-Noise Amplifiers (LNAs) as specific input ports into the RF reconfigurable array. Figure 51 depicts a cartoon block diagram of the new RASP 3.0 RF architecture [40].

A major application for the RASP 3.0 RF was to create delay lines using the inherent floating-gate switches used for signal routing. The RF array has large switch sizes optimized to reduce RF signal loss. A new S-Block switch design in the RF domain was created with the added functionality of a delay element. S-Block switches are designed to be chained together to create configurable delay lines. Delay lines are critical in the application of signal beamforming. This will enable the RF FPAA

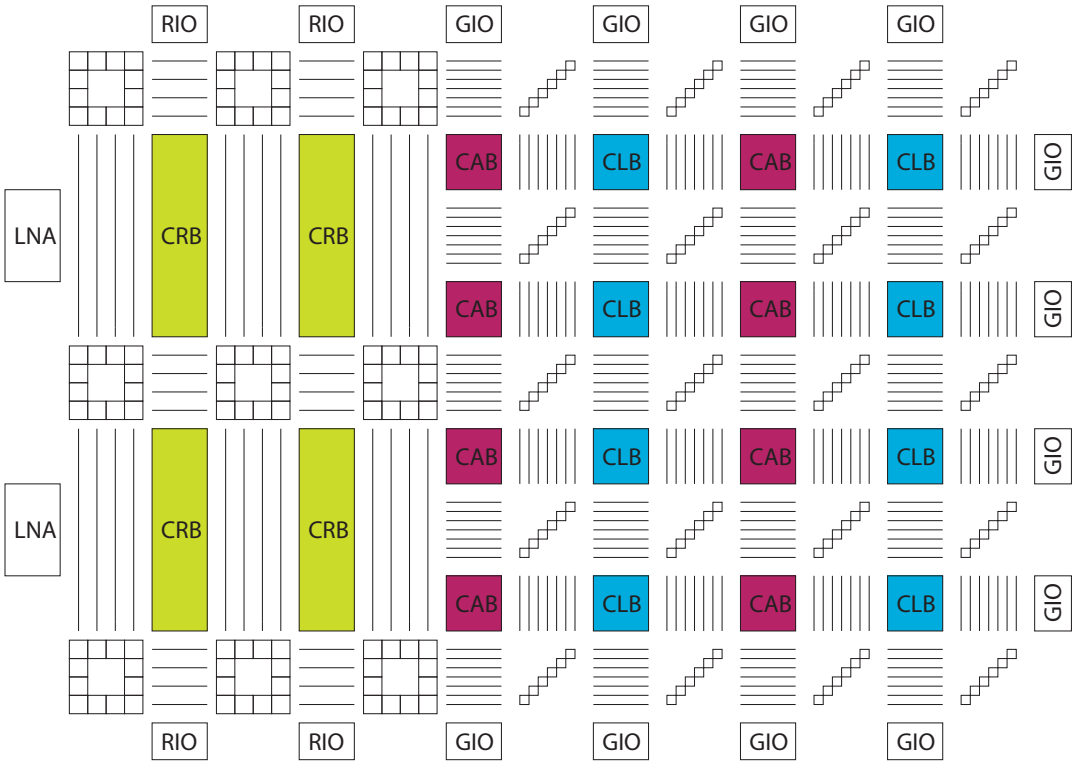


Figure 51: Conceptualized block diagram of the RASP 3.0 RF showing the RF front-end and baseband back-end.

to perform beamforming in the RF domain without the need for complex digital backends. Figure 52 diagrams a proof of concept delay line using the routing switches in the global interconnect of the RF FPAA [43].

For testing of the RASP 3.0 RF system, we created a mixed-mode RF test board shown in Fig. 53. The board has 7 matched line length 50 ω transmission lines feeding into the chip LNA inputs. We added a 1-4 GHz Local Oscillator (LO) generator with capability for internal or external signal generation. Similar to the RASP 3.0 test board, we added specialized power management and regulation for floating-gate programming, along with power management/regulation for normal system operation.

From testing the RASP 3.0 RF, we determined there was a systematic problem in the openMSP430 cpu. The chip in test was not able to communicate with a host PC using the built in 2-wire interface. We hypothesized the digital back-end of the RF

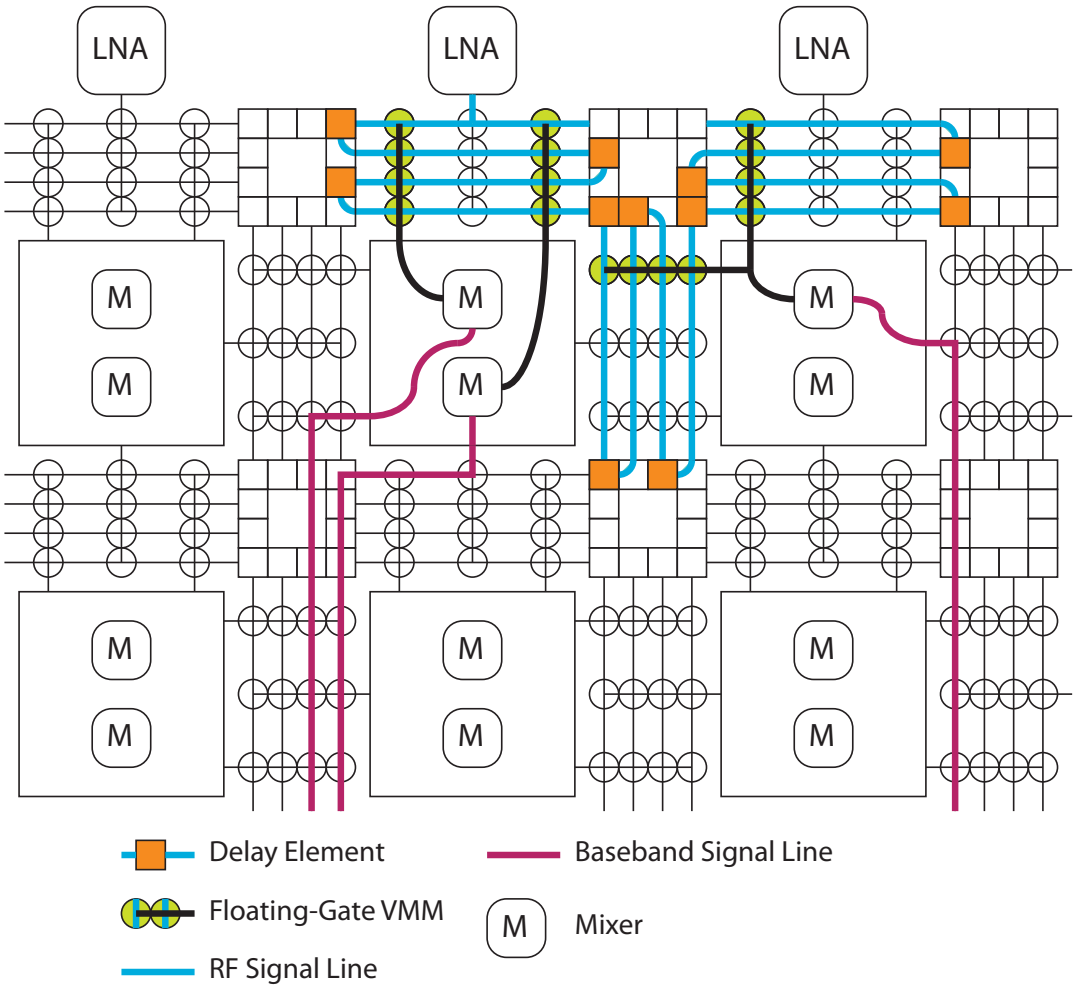


Figure 52: Conceptual diagram of delay lines created from the FPAA routing fabric.

FPAA had timing issues, i.e. setup and hold violations. After extensive debugging and analysis, we determined the synthesis and place-and-route information files utilized to build the digital backend infrastructure was flawed [43]. The information files were missing accurate parasitic values for the process metal interconnect. Also, we found incorrect timing checks within the verilog library files. With this knowledge, we fixed the issues and re-ran simulations of the CPU, targeting the serial interface. Figure 54 shows two simulation outputs. The top traces are with the original (incorrect) files and the bottom with the correct files. The bottom trace shows the serial interface being non-functional, while the top trace shows correct operation due to the incorrect design files. We can use the fixed design files to re-generate the digital backend with proper setup and hold times [43]. This will provide a design that will properly operate.

6.6 Conclusion

We have demonstrated a family of RASP chips which enable mixed-signal processing. Taking our results from the FPAADD, which was designed to integrate with the open source VTR/VPR routing tools and a modern global interconnect scheme, the RASP 3.0 added a digital back-end and volatile shift registers to the basic FPAADD design. Our toolset allows a system designer to create analog/digital/mixed-signal systems from a high level system-model viewpoint. Results from the RASP 3.0 verify chip functionality. The next step will be to use the RASP 3.0 to create larger systems and data converters from the work presented within. An RF optimized chip, the RASP 3.0 RF, was also built to leverage 40nm CMOS technology for faster reconfigurable mixed-signal systems. The progression of the RASP family has enabled more choice in the design of data converters and systems in general.

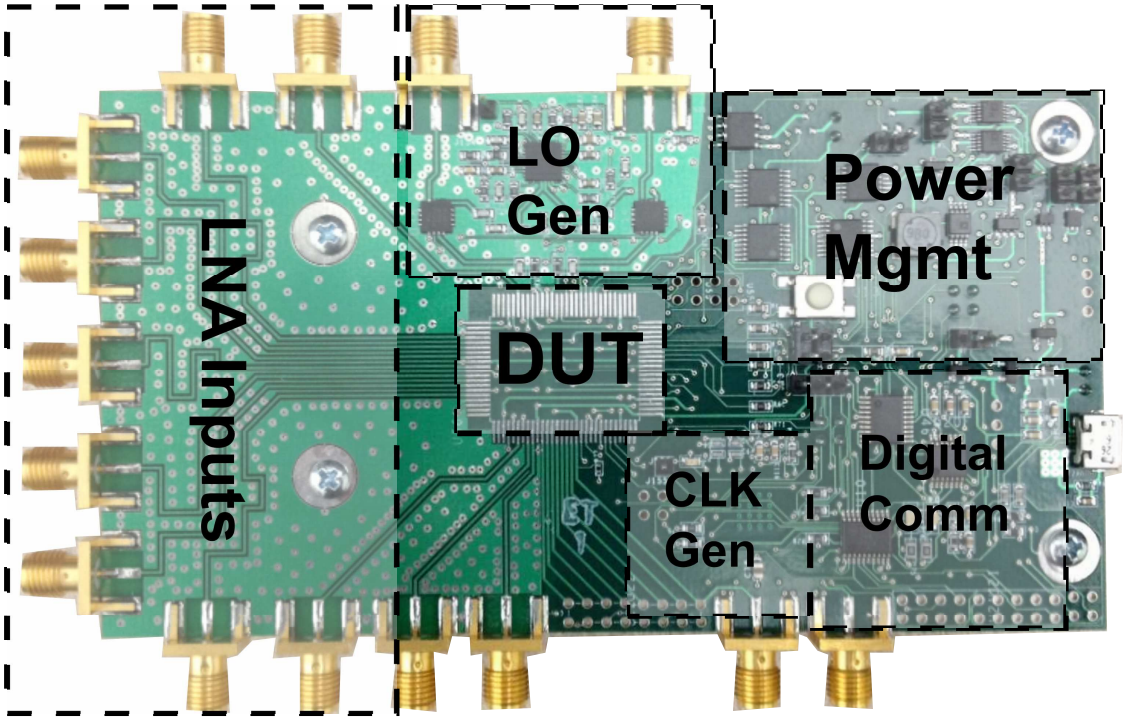


Figure 53: RASP 3.0 RF test board.

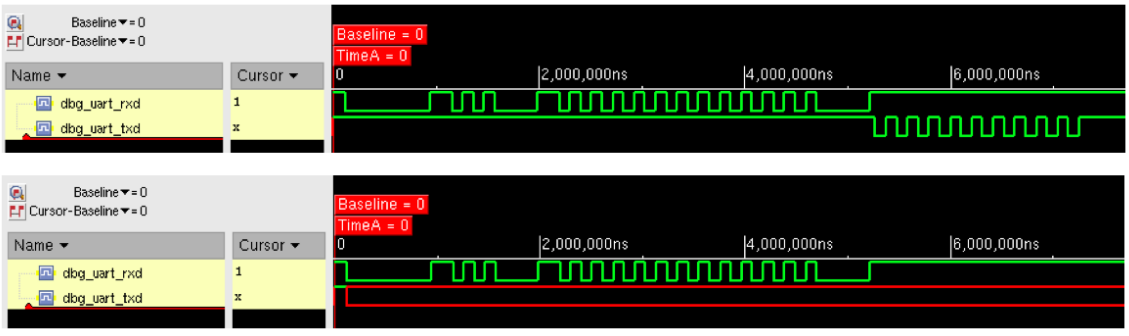


Figure 54: Digital simulation results using both accurate and inaccurate timing files.

CHAPTER VII

CONCLUSION

The subject of this research is the implementation and application of mixed-signal reconfigurable systems utilizing floating-gate transistors and field programmable analog/digital arrays. Towards this goal, various mixed-signal reconfigurable platforms have been designed, tested, and utilized to create circuits and systems. The basic charge holding behavior of floating-gate devices was exploited to enable further performance enhancements within the developed reconfigurable platforms. In this chapter, key research accomplishments and milestones that have been achieved in progressing towards the research goal will be summarized along with ideas for moving forwards in this area.

7.1 Research Summary

Chapter 2 provided an overview of floating-gate technology and the basics of programming the charge on a floating-gate transistor. We used two physical effects to modify the charge on a floating-gate transistor: electron tunneling and hot-electron injection. A basic method to programming multiple floating-gate devices was also described. This is a key step, as the ability to program more than one floating-gate transistor is required for large scale reconfigurable systems. We showed the basic system overview and implementation of array programming.

Chapter 3 introduced experimental results of floating-gate devices at technology nodes smaller than 350nm. Reconfigurable systems need to be able exploit the technological scaling afforded by CMOS technology. We have shown experimental data from a 40nm CMOS process of floating-gate transistors. These devices exhibit the ability to retain charge and modify charge via electron tunneling and hot-electron

injection. We have shown data displaying both physical effects and compared with data from a 350nm CMOS process. The results are comparable and provide sufficient cause for utilizing floating-gate transistors in very deep sub-micron nodes.

Chapter 4 introduced the use of floating-gate transistors to correct for analog errors and analog calibration. Transistors have a inherent mismatch in their characteristics due to minor differences in the actual geometry and fabrication of the devices. Floating-gate transistors provide a natural method to reduce the input-referred offset in matched transistors. We have implemented and shown data from differential pairs, current mirrors, and differential input amplifiers whose input-referred offset have been reduced. Also, we have shown the usage of floating-gates to tune/calibrate analog circuits such as filters and multipliers. Utilizing floating-gate transistors, we have created systems that can be tuned to the desires of the system users without needed DACs for each tuning point.

Chapter 5 introduced the FPAADD system and its improved mixed-signal architecture for FPAAs. We discussed the architecture in detail, including the change to a modern Manhanttan style global interconnect scheme. This allowed us to use standard and open-sourced routing tools for system design. The FPAAD also added digital CLB tiles into the array, giving the system FPGA capabilities. We have shown experimental data from the FPAADD that includes: basic system functionality, speed improvements due to architecture and buffered routing lines, and example data converter systems.

Chapter 6 introduced the RASP 3.0 chips. The RASP 3.0 chips are builtin upon the FPAADD and add a large digital back-end with volatile shift registers within the routing infrastructure. We have shown the design and implementation of the RASP 3.0 and the RASP 3.0 RF (an RF optimized version of the 3.0 chip). We have shown experimental data from the basic elements of the CAB and CLB devices of the RASP 3.0. The goal of the RASP 3.0 system is to enable greater variety of mixed-signal

systems including data converters.

7.2 List of Contributions

- Design, layout, and testing of floating-gate based circuits for offset removal.
- Design, layout, and testing of 40nm floating-gate test structures.
- Design, layout, and testing of floating-gate gilbert multiplier and G_m-C Filters. Collaborator: Ravi Chawl, and Guillermo Serrano.
- Design, layout, and testing of the FPAADD system. Collaborator: Richard Wunderlich.
- Design and layout of the following RASP 3.0 areas: CAB, Analog Memory and associated digital back-end, I/O blocks, and full-chip integration. Collaborators: Richard Wunderlich, Shubha Ramakrishnan, Suma George, Stephen Nease, Sam Shapero.
- Testing of the RASP 3.0 chip. Collaborators: Suma George, Sihwan Kim, Michelle Collins, Andrew Freeman, Sahil Shah.
- Design and testing of the RASP 3.0 test boards. Collaborators: Scott Koziol, and Stephen Nease.
- Design, layout, and testing of the RASP 3.0RF and associated RF test board. Collaborators: Richard Wunderlich, Stephen Nease, Taiyun Chi, Jong Seok Park.

REFERENCES

[1] Hasler, P., *Foundations of Learning in Analog VLSI*. PhD thesis, California Institute of Techonology, February 1997.

[2] Lee, K. F. E. and Gulak, P. G., "A transconductor-based field-programmable analog array," in *ISSCC Digest of Technical Papers*, pp. 198–199, Feb. 1995.

[3] Pankiewicz, B., Wojcikowski, M., Szczepanski, S., and Sun, Y., "A field programmable analog array for cmos continuous-time ota-c filter applications," *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 2, pp. 125–136, 2002.

[4] Becker, J., Henrici, F., Trendelenburg, S., Ortmanns, M., and Manoli, Y., "A field-programmable analog array of 55 digitally tunable otas in a hexagonal lattice," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 12, pp. 2759–2768, 2008.

[5] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C. M., and Hasler, P., "A floating-gate-based field-programmable analog array," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 9, pp. 1781–1794, 2010.

[6] *AN13x series AN23x Series AnadigmApex dpASP Family User Manual*.

[7] Fernández, D., Martínez-Alvarado, L., and Madrenas, J., "A translinear, log-domain fpaa on standard cmos technology," *Solid-State Circuits, IEEE Journal of*, no. 99, pp. 1–1, 2012.

[8] Hasler, P. and Lande, T., "Overview of floating-gate devices, circuits, and systems," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 1–3, Jan 2001.

[9] Hasler, P., "Continuous-time feedback in floating-gate mos circuits," *IEEE Transactions on Circuits and Systems II*, in Press.

[10] Bandyopadhyay, A., Hasler, P., and Anderson, D., "A cmos floating-gate matrix transform imager," *Sensors Journal, IEEE*, vol. 5, pp. 455–462, June 2005.

[11] Graham, D. W. and Hasler, P., "Capacitively-coupled current conveyer second-order section for continuous-time bandpass filtering and cochlea modeling," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Phoenix, Arizona), 2002.

[12] Ellis, R., Yoo, H., Graham, D. W., Anderson, D. V., and Hasler, P., "A continuous-time speech enhancement front-end for microphone inputs," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Phoenix, Arizona), 2002.

[13] Kucic, M., Low, A., Hasler, P., and Neff, J., "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 90–99, Jan 2001.

[14] Kucic, M., Dugger, J., Hasler, P., and Anderson, D., "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proceedings of the 21st Conference on Advanced Research in VLSI*, (Atlanta, GA), March 2001.

[15] Bandyopadyay, A., Serrano, G., and Hasler, P., "Programmaing analog computational elements to 0.2% accuracy over 3.5 decades using a predictive method," pp. 2148–2151, May 2005.

[16] Sze, S. M., *Physics of Semiconductor Devices*. Wiley-Interscience, 2006.

[17] Ashton, R., "Gate oxide thickness measurement using fowler-nordheim tunneling," in *Microelectronic Test Structures, 1991. ICMTS 1991. Proceedings of the 1991 International Conference on*, pp. 57–60, Mar 1990.

[18] Bandyopadhyay, A., Serrano, G., and Hasler, P., "Adaptive algorithm using hot-electron injection for programming analog computational memory elements within 0.2% of accuracy over 3.5 decades," *Solid-State Circuits, IEEE Journal of*, vol. 41, pp. 2107–2114, Sept 2006.

[19] Adil, F., Serrano, G., and Hasler, P., "Offset removal using floating gate circuits for mixed-signal systems," in *Southwest Symposium on Mixed-Signal Design*, pp. 190–195, Feb. 2003.

[20] Mead, C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[21] Chawla, R., Adil, F., Serrano, G., and Hasler, P., "Programmable gm ndash; c filters using floating-gate operational transconductance amplifiers," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 3, pp. 481–491, 2007.

[22] Monmasson, E. and Cirstea, M., "Fpga design methodology for industrial control systems - a review," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1824–1842, 2007.

[23] Hauck, S., "The roles of fpgas in reprogrammable systems," *Proceedings of the IEEE*, vol. 86, no. 4, pp. 615–638, 1998.

[24] Steiger, C., Walder, H., and Platzner, M., "Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks," *Computers, IEEE Transactions on*, vol. 53, no. 11, pp. 1393–1407, 2004.

[25] Wunderlich, R., Adil, F., and Hasler, P., "Floating gate-based field programmable mixed-signal array," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 8, pp. 1496–1505, 2013.

[26] Minch, B. A., Diorio, C., Hasler, P., and Mead, C. A., "Translinear circuits using subthreshold floating-gate MOS transistors," *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 167–179, 1996.

[27] Twigg, C. M. and Hasler, P., "A large-scale reconfigurable analog signal processor (rasp) ic," in *Proc. IEEE Custom Integrated Circuits Conf. CICC '06*, pp. 5–8, 2006.

[28] Hall, T. S., Twigg, C. M., Gray, J. D., Hasler, P., and Anderson, D. V., "Large-scale field-programmable analog arrays for analog signal processing," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 11, pp. 2298–2307, 2005.

[29] Vaughn Betz, Jonathan Rose, A. M., *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[30] Rose, J., El Gamal, A., and Sangiovanni-Vincentelli, A., "Architecture of field-programmable gate arrays," vol. 81, no. 7, pp. 1013–1029, 1993.

[31] B. Ahanin, S. V., "A high-density, high speed, array-based erasable programmable logic device with programmable speed/power optimization." ACM International Workshop on FPGA, February 1992.

[32] Leventis, P., Vest, B., Hutton, M., and Lewis, D., "Max ii: A low-cost, high-performance lut-based cpld," in *Proc. Custom Integrated Circuits Conf the IEEE 2004*, pp. 443–446, 2004.

[33] Hu, C., "Interconnect devices for field programmable gate array," in *Proc. Int. Electron Devices Meeting Technical Digest*, pp. 591–594, 1992.

[34] Baskaya, F., Reddy, S., Lim, S. K., and Anderson, D. V., "Placement for large-scale floating-gate field-programable analog arrays," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 8, pp. 906–910, 2006.

[35] Ganesan, S. and Vemuri, R., "Behavioral partitioning in the synthesis of mixed analog-digital systems," in *Proceedings of the 38th annual Design Automation Conference*, pp. 133–138, ACM, 2001.

[36] Jamieson, P. A. and Kent, K. B., "Odin ii: an open-source verilog hdl synthesis tool for fpga cad flows (abstract only)," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '10, (New York, NY, USA), pp. 288–288, ACM, 2010.

[37] "Berkeley logic synthesis and verification group, abc: A system for sequential synthesis and verification, release 70731. http://www.eecs.berkeley.edu/ alanmi/abc/."

[38] Luu, J., Kuon, I., Jamieson, P., Campbell, T., Ye, A., Fang, W., and Rose, J., "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 133–142, ACM, 2009.

[39] Schlottmann, C., Shapero, S., Nease, S., and Hasler, P., "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *Solid-State Circuits, IEEE Journal of*, vol. 47, pp. 2174–2184, Sept 2012.

[40] Wunderlich, R. B., *Floating-Gate-Programmable and Reconfigurable, Digital and Mixed-Signal Systems*. PhD thesis, Georgia Institute of Technology, 2014.

[41] Betz, V. and Rose, J., "Vpr: A new packing, placement and routing tool for fpga research," in *FPL* (Luk, W., Cheung, P. Y. K., and Glesner, M., eds.), vol. 1304 of *Lecture Notes in Computer Science*, pp. 213–222, Springer, 1997.

[42] Aziz, A., Balarin, F., Cheng, S.-T., Hojati, R., Kam, T., Krishnan, S., Ranjan, R., Shiple, T., Singhal, V., Tasiran, S., Wang, H.-Y., Brayton, R., and Sangiovanni-Vincentelli, A., "Hsis: A bdd-based environment for formal verification," in *Design Automation, 1994. 31st Conference on*, pp. 454–459, June 1994.

[43] Nease, S. H., *Neural and Analog Computation on Reconfigurable Mixed-Signal Platforms*. PhD thesis, Georgia Institute of Technology, 2014.

VITA

Farhan Adil was born in Dhaka, Bangladesh. He received his B.S. in Biomedical Engineering from Johns Hopkins University in 2001 and his M.S. in Electrical and Computer Engineering from Georgia Institute of Technology in 2003. From 2003-2007, he was with Northrop Grumman Electronic Systems working on data converters for radar receivers and other sensing platforms. He received his Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology in 2014. His research interests include low-power analog signal processing, high performance data converters, and mixed-signal IC design.